uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Electrónica Industrial y Automática
2017-2018

*Trabajo Fin de Grado*

# "Integración de visión artificial en drones multi-rotor para gestión de misiones autónomas"

---


Mohammed Bouayad Boughroum

Tutor

Jesús García Herrero


Leganés, 2018

# RESUMEN

La presente memoria corresponde al diseño de un sistema de vigilancia para recintos privados a través del uso de un dron comercial en combinación con una computadora de bajo coste y su correspondiente cámara para adquisición de imágenes.
El objetivo que se pretende alcanzar con este proyecto es cargar una misión autónoma al dron y en función de los eventos del entorno adquiridos, a través de la cámara, dotar al dron de la capacidad de manipular la trayectoria predefinida por el usuario, sea cual sea, la trayectoria definida.
En concreto, se trata de un sistema formado por la controladora de vuelo del dron (Pixhawk) en adicción a una Raspberry Pi que actúa como elemento de cómputo. La Raspberry Pi permite al sistema analizar imágenes en tiempo real, buscando en cada momento que no haya presencia de objetos intrusos en el recinto a vigilar.
Gracias al sistema implementado, obviamente, estableciendo una serie de requisitos a nivel de seguridad e iluminación, es posible conseguir detectar la presencia de vehículos, caras y personas. Una vez realizada la detección, el dron realizará una maniobra de acercamiento para tomar constancia del objeto intruso a través de una foto o de un video, permitiendo, un posterior análisis por parte del propietario del recinto o autoridades pertinentes.

**Palabras clave:**
Vigilancia; Dron; Raspberry Pi; Pixhawk; OpenCV

# DEDICATORIA

Me gustaría agradecer a todas las personas que me han ayudado y animado a la elaboración de este trabajo. Particularmente, me gustaría agradecer a mi familia y a los 4 compañeros con quienes he compartido momentos desde mi primera hasta mi última clase en la universidad; compañeros, amigos este trabajo va por vosotros.

Finalmente me gustaría agradecer a Jesús García Herrero por su paciencia y compromiso. Gracias Jesús, has sido un tutor impecable.

# INDICE DE CONTENIDOS

# INDICE DE FIGURAS

# INDICE DE TABLAS

# 1 INTRODUCCIÓN

## 1.1. Motivación para elaborar Trabajo

Ante la creciente expansión tecnológica en el sector de los drones se ha considerado que un análisis profundo de este sector puede proporcionar grandes aplicaciones prácticas a nivel de usuario. En este sentido, con el paso del tiempo, los drones se van haciendo un hueco en la sociedad y cada vez son más los usuarios los que ensalzan sus características y aceptan su incorporación de manera masiva a la sociedad.

En este trabajo se busca dotar de utilidad a estos vehículos no tripulados con la intención de extraer nuevas características y obtener funcionalidades que sean de provecho para el día a día.

Siguiendo esta línea, se ha considerado que una de las aplicaciones más interesantes a implementar se corresponde con el uso de drones como sistemas de vigilancia. No se trata de un área de estudio que haya aparecido hace relativamente poco, pues bien, hace ya años que aparecieron trabajos y proyectos relacionados con esta área en busca de mejorar los existentes sistemas de vigilancia.

Son numerosos los sectores en los que se ha buscado establecer a los drones como sistema de vigilancia alternativo: vigilancia de aeropuertos, campos de cultivo, invernaderos, infraestructuras ferroviarias o vigilancia de tráfico son algunos de los ejemplos donde se puede aplicar este tipo de sistemas.

## 1.2. Objetivo

El principal objetivo de este proyecto es conseguir, a través del uso de elementos de bajo coste, diseñar un sistema de detección y seguimiento de objetos y posterior grabación o captura de los mismos mediante el uso de un dron comercial en una zona controlada ante cambios de iluminación y estableciendo las precauciones de seguridad pertinentes para mantener la integridad física del dron.

Para alcanzar el objetivo mencionado, se embarca en el chasis del dron el sistema compuesto por la controladora de vuelo (Pixhawk) y la microcomputadora Raspberry Pi. Combinando estos dos dispositivos se consigue establecer un lazo de comunicación entre ambos que permite intercambiar información precisa, dando posibilidad de conocer, en todo momento, la información proveniente de los sensores integrados en el dron y la información que proviene de la Raspberry Pi, así como, la posibilidad de actuar sobre los diferentes rotores de los que dispone el dron.

Para diseñar de manera efectiva el sistema de vigilancia definido se deben llevar a cabo las siguientes tareas:

- Diseño de un sistema de visión fiable.
- Diseño de un sistema de control y manipulación fiable.
- Integración de ambos diseño.

A lo largo de la memoria se explicará con mayor nivel de detalle todos los pasos seguidos para la obtención del sistema deseado.

### 1.3 Marco Regulador

En lo que a normativa para vehículos aéreos no tripulados se refiere, se puede decir que, a pesar de que se está trabajando en una nueva legislación acorde en todos los países de Europa, lo cierto es que, actualmente, cada país dispone de sus propias leyes para el tratamiento del uso de drones.

Puesto que el diseño del sistema planteado se va a implementar en un dron que se vuela en España, se ha decidido analizar la normativa aplicable en este sector únicamente en España. En este sentido, es en 2014 cuando en España se aprueba el primer marco regulador en cuanto a actividades relacionadas con drones se refiere.

El pasado 29 de diciembre de 2017 se hace público en el boletín oficial del estado (BOE) el nuevo marco normativo que regula el uso civil de aeronaves pilotadas por control remoto [16].

Con la nueva legislación se busca eliminar, en cierto sentido, gran parte de las restricciones existentes previas al mismo, restricciones como vuelos nocturnos, vuelos por zonas con altas aglomeraciones de personas, etc. Con esto, aparecen nuevos escenarios a los que se busca analizar y dotar de una solución.

Antes del nuevo reglamento, el único escenario legal de operación era el vuelo diurno y en zonas rurales exteriores a las ciudades. Gracias al nuevo decreto y cumpliendo en todo momento los requisitos de seguridad establecidos por la agencia estatal de seguridad aérea (AESA) se ha ampliado los escenarios operativos.

Poder volar por espacios aéreos controlados, integración en actividades urbanas, control de tráfico o vuelos nocturnos son algunos de los escenarios que permite la legislación vigente. Ahora bien, como condición general, para poder volar drones en entornos urbanos se requiere que los drones no alcancen alturas superiores a 120 metros y que su peso sea inferior a 10 kilogramos.

Para el caso de usuarios aficionados la normativa vigente no permite realizar vuelos en entornos urbanos, a menos que, el peso del dron sea inferior a 250 gramos, de la misma manera, solo se permiten vuelos diurnos y en unas condiciones meteorológicas favorables. También, se restringe a más de 8 kilómetros la distancia que debe separar el lugar donde se vuela el dron al aeropuerto más cercano. Estas condiciones buscan garantizar la seguridad de las personas, así como, los bienes materiales que se hallan en zonas urbanas.

### 1.4. Estructura del documento

El documento se organiza de tal manera que se busca dar a conocer todos los pasos seguidos en la elaboración del proyecto. En este sentido, primero se da una definición del algunos de los conceptos clave, posteriormente, se analizan todos los métodos para alcanzar la solución buscada. Seguidamente, se muestran los resultados obtenidos, en un entorno simulado del sistema diseñado, y finalmente se plantean posibles mejoras y conclusiones del trabajo desarrollado.

# 2 ESTADO DEL ARTE

## 2.1 Conceptos Previos

Antes de acceder a realizar un estudio exhaustivo del proyecto que se aborda, es necesario conocer algunos conceptos básicos.

### 2.1.1 Dron (UAV)

Un concepto importante en este proyecto, es el concepto de dron. La definición más adoptada y extendida para este término es la que hace referencia a un vehículo aéreo no tripulado (UAV). Ahora bien, pueden aparecer otras definiciones en función de la fuente escogida.

En la actualidad, estos vehículos no tripulados están experimentando avances muy importantes, y es que, es un campo de investigación que despierta gran interés tanto para profesionales del sector como para usuarios aficionados.

**Figura 1. Dron (UAV) Hexacoptero [17]**

Los drones disponen de una serie de características que los hacen ser unos de los dispositivos más atractivos y demandados por los amantes de la tecnología:

- Coste competitivo
- Alta potencia
- Control simple y sencillo
- Equilibrio en el aire
- Etc…

Ante la creciente demanda de UAVs, comercialmente están aparecido gran cantidad de configuraciones en cuanto a tamaño, peso o número de motores (4 hasta 16). Las diferentes configuraciones, otorgan mayor versatilidad para abordar proyectos de diferentes características y especificaciones. En este sentido, el campo de aplicación de estos vehículos es muy amplio y variado. Se ha introducido este tipo de dispositivos a tares médicas, seguridad, cinematografía, etc.

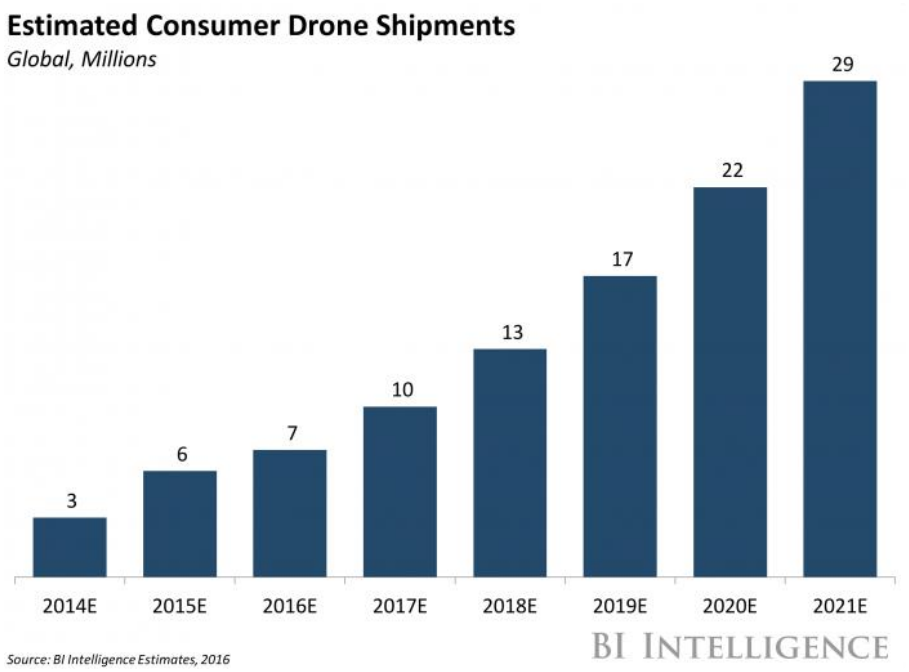


**Figura 2. Demanda anual de drones [13]**

### 2.1.2 Librería OpenCV

OpenCV es una librería de libre acceso orientada al procesamiento de imágenes y al aprendizaje automático. Se trata de un software desarrollado en el año 1999 por Intel y que dispone de más de 2500 algoritmos optimizados.

OpenCV es una librería que destaca por su gran versatilidad, pues es una librería que es compatible con una gran cantidad de sistemas operativos y lenguajes de programación. En este sentido, soporta sistemas operativos como Windows, Linux, Mac OS o Android. También, es compatible con lenguajes de programación tales como C/C++, Python, Java o Matlab.

OpenCV tiene una comunidad de usuarios de más de 47.000 usuarios y se estima que el número de descargas de esta librería supera los 14 millones.

En lo que respecta al proyecto que se trata, la Raspberry Pi cuenta por defecto con la versión 2.4.9 de OpenCV y lenguaje de programación Python. En este proyecto, se ha decidido comparar el rendimiento y soluciones que pueden llegar a aportar tanto el lenguaje Python como lenguaje C++.En relación a esto, se ha decidido descargar e instalar una versión reciente de OpenCV (versión 3.0.0). Esta versión cuenta con algoritmos optimizados y gran cantidad documentación de referencia.

En futuros apartados de esta memoria se darán más detalles acerca del rendimiento obtenido para el algoritmo de detección utilizando ambos lenguajes de programación mencionados.

### 2.1.3 Raspberry Pi 3 modelo B

La Raspberry Pi es una computadora de placa simple (SBC), diseñada y construida con el objetivo principal de hacer llegar a todo el mundo un sistema que otorgue la posibilidad de acceder a herramientas informáticas sin necesidad de disponer de un ordenador convencional. En este sentido, la principal aplicación a la que se ha asociado ha sido a la educación.

**Figura 3. Raspberry Pi 3 modelo B [2]**

La Raspberry Pi 3 modelo B proporciona unas características bastantes interesantes a un precio bastante asequible. Se trata de una herramienta con grandes funcionalidades, por lo es ideal para la elaboración de una amplia gama de proyectos dispares unos de otros.

Es un dispositivo electrónico de bajo consumo y se corresponde con la tercera generación de estos dispositivos Raspberry Pi. Como características técnicas principales de estos aparatos se puede destacar:

- CPU ARMv8 a 1.2GHz de 64-bit con 4 núcleos
- 1 GB de RAM
- Bluetooth 4.1
- WIFI 802.11 b/g/n
- 40 pines entradas-salidas digitales(GPIO)
- 4 puertos USB
- Puerto HDMI
- Puerto CSI para cámara Raspberry Pi
- Puerto DSI para pantalla Raspberry Pi
- Ranura MicroSD
- Puerto Jack
- Puerto MicroUSB para alimentación
- Puerto Ethernet Cable RJ-45

La Raspberry Pi no dispone de memoria interna por defecto, sino que lleva asociado un "*slot"* para tarjeta MicroSD donde se almacenará e instalará el sistema operativo y toda la información que almacene la placa electrónica. Aunque es recomendable que la

tarjeta SD tenga la mayor capacidad posible, para este proyecto se dispone de una tarjeta SD de 8 GB de capacidad.

La mayor utilidad que se le está dando a la Raspberry pi se enfoca en conseguir crear algo lo más parecido posible a un ordenador convencional. Pues bien, para conseguir crear una computadora lo más parecido a un ordenador convencional, con este dispositivo electrónico, basta con descargar e instalar un sistema operativo en la SD, colocar los diferentes periféricos en los diferentes puertos USB como son el ratón o el teclado. A través del puerto HDMI es posible conectar con un monitor.

Por otro lado, es posible añadir elementos extra como pueden ser altavoces o equipos de sonido mediante el puerto JACK o establecer una conexión a internet ya sea por medio de conexión WIFI, gracias a un adaptador WIFI, ó conexión por cable por medio de Ethernet (cable RJ45).

Aunque no se ha comentado, resulta evidente que el dispositivo requiere de alimentación eléctrica, esto se hará mediante un cargador estándar. Ahora bien se recomienda que el cargador dé como salida una tensión de 5V y corriente de 2,5A.El consumo de la Raspberry Pi es realmente bajo, pues consume poco menos de 5W.

Más adelante se comentará con más detalle todos los pasos para configurar de manera adecuada la Raspberry Pi.

### 2.1.4 Pixhawk

En el diseño, tanto mecánico como electrónico, de un dron cualquiera es de gran importancia definir y establecer una controladora de vuelo fiable, pues bien, es está la que se encarga de controlar todos los movimientos del dron. La controladora de vuelo procesa toda la información recibida de los diferentes sensores integrados en el dron y proporcionar un control adecuado a las salidas o rotores del dron.

La controladora de vuelo, Pixhawk, es una pequeña microcomputadora provista de una serie de sensores de estabilidad, posicionamiento GPS y transferencia de información por telemetría. Se puede decir que se trata de una de las mejores controladoras de vuelo que se encuentran en el mercado, es una evolución natural de Ardupilot y Arduino, y que su condición de hardware libre lo convierte en un dispositivo electrónico altamente usado en el campo de drones tanto en investigaciones profesionales como en actividades de ocio.

Una de las ventajas de estas controladoras de vuelo, y que en este proyecto es de vital importancia, es que ofrece la posibilidad de establecer una conexión con la Raspberry Pi a través del puerto de telemetría secundario.

A continuación, se va a proceder a realizar un estudio de las características principales de las controladoras Pixhawk y posteriormente se mostrará los diferentes pines y puertos que dispone esta controladora.

En cuanto a las características del procesador, se puede decir que la controladora dispone de un microprocesador ARM de 32 bits *"Cortex M4 core"* con FPU a 168

MHz, una memoria RAM de 256 KB y 2 MB de flash. En caso de fallo del microprocesador principal se dispone de un coprocesador también de 32 bits.

En lo que alimentación de la placa se refiere, se puede usar la tensión que se obtiene de los ESC o usar una tensión exterior, ambas conectadas al puerto "Power" de la controladora. Por último, las pequeñas dimensiones del dispositivo (81.5 mm longitud, 50 mm anchura y 15.5 mm de altura) y su reducido tamaño lo hacen ideal como elemento embarcado.


**Figura 4. Pixhawk Puertos frontales [3]**

En la imagen anterior, se muestra una vista en planta de todos los puertos de los que dispone la Pixhawk. Se han señalado todos los puertos así como una pequeña descripción de los mismos.


**Figura 5. Pixhawk - Puertos Laterales [3]**

En la imagen anterior, se presenta una vista de los puertos laterales de la Pixhawk. También se da una pequeña descripción de cada uno de ellos. A continuación se pueden ver los diferentes pines de salida de los que está formada la controladora de vuelo.

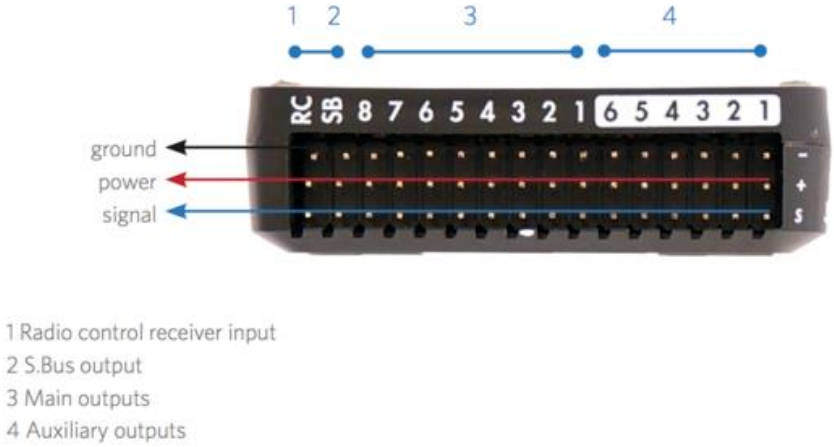


**Figura 6. Pixhawk Pines de entrada-salida**

### 2.1.5 MAVLink ("Micro Air Vehicle Link")

MAVLink es un protocolo de comunicaciones diseñado para establecer una comunicación directa con vehículos aéreos no tripulados. Este protocolo data de 2009 y puesto en marcha por Lorenz Meier.

Este protocolo es ideal para transmitir datos como velocidad, orientación y ubicación del dron o vehículo. Este sentido es el protocolo ideal para establecer una comunicación con la estación en tierra y el vehículo en cuestión. La tramad del paquete se divide en 8 campos.

MAVLink es fácilmente integrable en diversidad de vehículos y estaciones en tierra. Por otro lado, además de la ingente cantidad de mensajes predefinidos de los que dispone es posible crear nuevos mensajes de forma personalizada.

| Inicio | Longitud | NºSec | ID Sist. | ID comp. | ID Mens. | Payload | Checksum |
|---|---|---|---|---|---|---|---|

**Tabla 1. Paquete MAVLink**

Cada uno de los seis primeros campos del paquete se corresponde con un byte. Estos seis campos forman lo que se conoce como cabecera:

- **Inicio:** Indica el comienzo del paquete
- **Longitud:** Indica el número de bytes que se envían en el campo Payload. Toma valores entre 0 y 255.
- **Nº Secuencia:** Muestra el número de paquete enviado.
- **ID del sistema:** Identifica el sistema para diferenciar entre varias de ellos.
- **ID mensaje:** Indica el modo de lectura de Payload.

Los otros campos son:

- **Payload** del paquete que es donde se almacena la información a transmitir y cuya longitud se especifica en la cabecera.
- **Checksum** del paquete permite controlar errores que puedan aparecer.

# 3 MÉTODOS DE IMPLEMENTACIÓN DEL SISTEMA

En este apartado de la memoria se va a hacer un análisis de los diferentes métodos existentes que permitan desarrollar un sistema que combine la información proveniente de la Raspberry Pi con la controladora de vuelo Pixhawk, así como, escoger la aplicación que permita crear y visualizar las misiones desde una estación de tierra.

## 3.1 ROS || DroneKit-Python

Para conseguir la interacción entre la Pixhawk y la Raspberry Pi, y así poder enviar comandos desde la microcomputadora a la controladora de vuelo pueden usarse dos vías:

- ROS ("Robotic Operating System")
- DroneKit-Python API

Corriendo uno de estos dos sistemas en la Raspberry Pi, es posible dotar al conjunto Pixhawk+ Raspberry Pi de un control flexible sobre la unidad de control de vuelo, así como, la capacidad de añadir funcionalidades extra como puede ser visión artificial.

**ROS ("Robo tic Operating System")**

ROS es un marco flexible para desarrollar software para robots. Se trata de una colección de bibliotecas, herramientas y convenios con los que se pretende simplificar la tarea de crear comportamientos complejos y robustos en una amplia variedad de plataformas robóticas.

ROS no es un sistema operativo por sí mismo, sino que, trabaja con otros sistemas operativos para ofrecer nuevos servicios a la hora de desarrollar software para robots. En este sentido, se debe aclarar que ROS no es un lenguaje de programación a pesar de que este escrito por defecto en C++.

ROS es compatible con ciertas placas de sistemas empotrados como son:

- Raspberry Pi (modelo B, Zero)
- Odroids
- BeagleBone
- Nvidia Jetson TK1

Por otro lado, también se puede decir que ROS es compatible con varios sistemas operativos como son: Ubuntu, Debian, Gentoo o Android NDK.

Por último, existen varias distribuciones ROS (Lunar, Kinetic, indigo, jade...). La página oficial de ROS recomienda el uso de la distribución Kinetic que es una distribución compatible con el sistema operativo oficial de Raspberry Pi, *Raspbian Jessie.*

El hecho de añadir un elemento adicional embarcado en el dron (Raspberry Pi) dotado de ROS otorga una serie de beneficios como puede ser el aumento de velocidad en el procesamiento de procesos, así como la adición de una serie de sensores y actuadores soportados por la comunidad ROS.

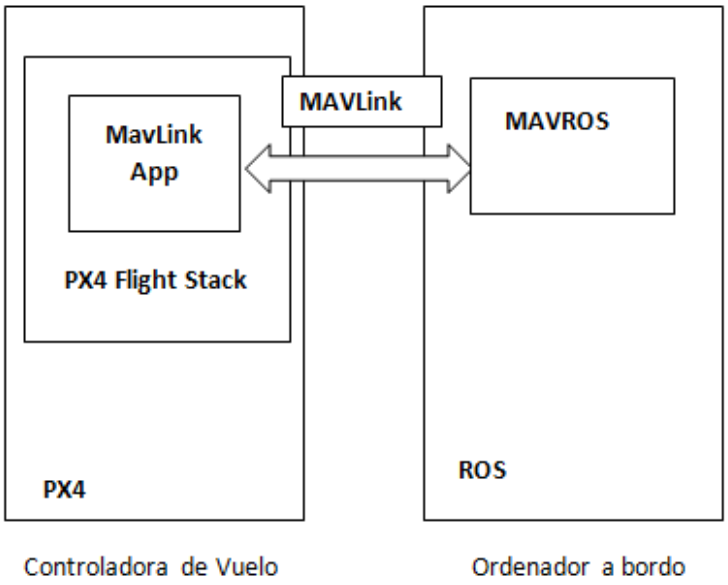


**Figura 7. Comunicación entre Pixhawk y ROS**

MAVROS es un paquete ROS que permite la comunicación entre diferentes dispositivos que corren ROS, autopilotos o estaciones en tierra con MAVLink activado. En este sentido, MAVROS se puede utilizar para comunicar una controladora de vuelo cualquiera con un ordenador con ROS habilitado.

En definitiva, MAVROS es el puente de unión entre ROS y el protocolo MAVLink.

**DroneKit-Python**

DroneKit-Python es una herramienta que posibilita la creación de aplicaciones muy variadas con un alto grado de robustez para una alta gama de UAVs.

Como se ha comentado anteriormente, es una herramienta que se corre en un ordenador a bordo del dron, en este caso Raspberry Pi, complementando a las acciones de la controladora de vuelo para gestionar información computacional en masa y para llevar a cabo aplicaciones que requieran una latencia o un tiempo de respuesta lo más rápido posible. Es decir, con la Raspberry Pi corriendo DroneKit API se consigue alcanzar mayor inteligencia y versatilidad a la que solo la controladora de vuelo puede ofrecer.

Esta API se comunica con los vehículos, y por ende, con la controladora de vuelo a través de MAVLink, esto permite conocer toda la información acerca de estados o parámetros dinámicos del dron, así como, la gestión de misiones y control de movimientos. En este sentido, todas las controladoras de vuelo que se conectan por medio de MAVLink son compatibles con DroneKit.

Se trata de una herramienta de libre acceso (“Open Source”), por lo que es posible encontrar soluciones a problemas que puedan aparecer en foros y páginas web de ayuda creados por la gran comunidad de usuarios que hay detrás.

Por otra parte, las computadoras que son compatibles con DroneKit y a su vez con Ardupilot son:

- Raspberry Pi

- Intel Edison
- Odroid
- BeagleBoneBlack

El uso de DroneKit, como herramienta de control, aporta un gran abanico de funcionalidades, esto es, entre otras cosas permite:

- Establecer conexión con un vehículo desde un código fuente
- Monitorizar y ajustar información relacionada con el dron.
- Controlar el movimiento del dron a través del modo guiado
- Crear y manipular misiones en modo autónomo

Por último, DroneKit es compatible con gran cantidad de sistemas operativos, ya sea Linux, Windows o Mac OS X. La instalación es muy sencilla, para Linux se deben llevar a cabo los siguientes pasos:

1. Instalar Python-dev y pip
    - *Sudo apt-get install python-pip python-dev*
2. Instalar DroneKit
    - *Pip install dronekit*
3. Instalar simulador para DroneKit, DroneKit-SITL
    - *Pip install dronekit-sitl*

## 3.2 QGroundControl

Cuando se habla de drones, se busca disponer de un medio a partir del cual monitorizar en todo momento características dinámicas del mismo obtenidas a partir de telemetría, así como, disponer de la posibilidad de controlar el vuelo del UAV. Para llevar a cabo esta tarea existen varias aplicaciones que actúan como estación en tierra, aunque en este caso se va a especial énfasis en QGroundControl.

QGroundControl proporciona un control completo para vehículos con plataformas PX4 o Ardupilot. Es una aplicación que proporciona una interfaz y un uso sencillo para principiantes al mismo tiempo que pone en servicio funciones complejas para usuarios más experimentados. QGroundControl como estación en tierra ofrece las siguientes características:

- Posibilidad de ajustar y configurar vehículos dotados de PX4 o Ardupilot.
- Control de vuelo para vehículos dotados de PX4, Ardupilot o cualquier otra controladora de vuelo que se comunique a través de MAVLink
- Creación de misiones para modo autónomo.
- Mapeado de trayectoria de vuelo y monitorización datos dinámicos.
- Gestión de varios vehículos
- Grabación de Video.

- Compatibilidad con Windows, OS X, Linux, iOS o Android.

Poner en marcha QGroundControl es un proceso bastante sencillo. En caso de Linux, primero se descarga el archivo con extensión. AppImage. Posteriormente se instala el archivo descargado pinchando 2 veces o a través la terminal de comandos:

- *Chmod +x ./QGroundControl.AppImage*
- *./QGroundControl.AppImage*

Finalmente, se conecta al vehículo a la estación en tierra vía USB, telemetría o WiFi. La detección y conexión al vehículo se hace de manera automática.

### 3.3 Python || C++

Al inicio de cualquier proyecto surge la duda de que lenguaje de programación utilizar para abordar el mismo. En este proyecto se debe elaborar un código fuente que permita relacionar diferentes librerías, en este caso una librería para análisis y manipulación de imágenes y otra librería que permita intercambiar comandos con la controladora de vuelo en función de los resultados obtenidos en el análisis de las imágenes.

Para tal situación, en apartados anteriores, se pusieron en liza los diferentes métodos para abordar el proyecto. En este sentido, se ha planteado el uso de ROS con C++ o DroneKit con Python como herramientas y lenguaje global para el diseño del sistema de gestión de misiones autónomas junto con el sistema de visión.

Con todo lo dicho, para abordar el proyecto, se ponen en juego dos lenguajes de programación: Python y C++


**Figura 8. Logo Lenguaje Python [21]**

Python es un lenguaje de programación creado en 1989 en manos de Guido Van Rossum. Es considerado un lenguaje interpretado, de alto nivel, multiparadigma, tipado dinámico y multiplataforma. Cuando se habla de lenguaje interpretado se hace referencia que se trata de un lenguaje que no requiere de un compilador sino un intérprete para ejecutarse.

Es un lenguaje de alto nivel, pues, es un lenguaje con una estructura sintáctica y semántica legible, independiente de la arquitectura hardware por lo que dispone de mayor grado de portabilidad. Multiparadigma indica que es un lenguaje que acepta diferentes técnicas de programación: funcional, orientada a objetos o de aspectos. Además, es un lenguaje multiplataforma por lo que puede ser interpretado en diferentes sistemas operativos.

Por último, es un lenguaje de tipado dinámico, lo que indica que no se requiere definir variables asignando el tipo de dato que es, sino que se hace de manera automática en tiempo de ejecución.

**Figura 9. Logo Lenguaje C++[22]**

C++ es un lenguaje de programación que aparece en 1979 en manos de Bjarne Stroustrup. Se trata de una extensión de C donde se busca la creación y manipulación de objetos. Al igual que Python es un lenguaje de alto nivel, multiparadigma multiplataforma. Se diferencia de Python por ser un lenguaje compilado, es decir, ejecuta el programa generando previamente un ejecutable y por no ser un lenguaje de tipado dinámico, puesto que, al definir cualquier variable se debe especificar el tipo de dato al que pertenece.

En definitiva, se puede decir que tanto Python como C++ son 2 lenguajes de programación con altas prestaciones y funcionalidades y que en función del proyecto que se desee desarrollar será más conveniente uno u otro. En este proyecto, en particular, se ha decidido hacer un análisis de ambos lenguajes en lo que a procesamiento de imágenes se refiere probando cual de ambos extrae mejores resultados.

Ahora bien, en cuanto al diseño del algoritmo completo para la gestión de misiones autónomas se ha considerado definir Python como lenguaje de programación, por su sencillez en cuanto a análisis e interpretación se refiere, expresividad, así como, la posibilidad que ofrece de ejecutar varias partes del algoritmo de forma simultánea En consecuencia con lo anterior, se ha considerado utilizar DroneKit- Python API como sistema que permita mandar comandos entre la Raspberry Pi y la Pixhawk y QGroundControl como estación en Tierra.

# 4 DISEÑO ALGORITMO DE VISIÓN

En este apartado, se va explicar todos los procedimientos seguidos para conseguir la elaboración de un algoritmo de detección de los objetos de interés usando la librería OpenCV.

En primer lugar, se va explicar la puesta en marcha de la Raspberry Pi y la posterior descarga de las librerías, así como, los diferentes métodos de detección usados para la detección de cada uno de los objetos de interés.

## 4.1 Configuración y puesta en marcha de la Raspberry Pi 3

### 4.1.1 Instalación del sistema operativo a la Raspberry Pi

En este primer apartado, el objetivo es instalar un sistema operativo a la Raspberry Pi. Se ha decidido instalar la distribución Raspbian de GNU/Linux basado en Debian, que es la distribución original de Raspberry Pi.

Lo primero que se debe hacer es formatear la tarjeta SD donde se vaya a almacenar el sistema operativo, en el caso a tratar se va usar una tarjeta MicroSD de 8GB para eliminar cualquier tipo de elemento o archivo basura que tenga por defecto la tarjeta de almacenamiento.

El siguiente paso es descargar la imagen .iso de la última versión del sistema operativo Raspbian a través de la página web oficial de Raspberry Pi [5].


**Figura 10. Logo Sistema Operativo Raspbian**

Por último, usando cualquier programa de los múltiples que se puede encontrar en la red, se quema la imagen .iso en la tarjeta de almacenamiento.

### 4.1.2 Conexión remota con Pc

La Raspberry Pi, como cualquier otro tipo de computador, requiere de una serie de periféricos para el control y el manejo de la misma. Ahora bien, para abordar el objetivo que se pretende conseguir, no tiene mucho sentido el empleo de estos periféricos. Pues bien, la computadora se va a embarcar junto a la controladora Pixhawk y es de vital importancia poder controlarla remotamente desde un punto fijo. Existen 2 métodos para realizar este control remoto, usando protocolo SSH por lo que control se hace de modo textual, o bien usando un entorno gráfico. Para este proyecto se ha decidido emplear como medio de conexión remota el uso de un entorno gráfico.

Mediante el empleo de un ordenador portátil convencional, se puede establecer una conexión con la Raspberry Pi de manera sencilla estableciendo una conexión cliente-servidor. En este sentido, basta con poner en marcha el servidor de la Raspberry Pi y acceder a él desde el cliente descargado en el portátil.

Para activar el servidor de la Raspberry pi, se puede hacer dos maneras, tanto gráficamente como desde la consola de comandos.

En el primer caso, se accede a la ruta que se apunta a continuación y activar la opción VNC.

*Inicio>>>Preferencias>>>Configuración Raspberry Pi>>>Interfaces*

Por otro lado, se puede emplear la terminal de comandos. Usando el comando:

- *Sudo raspi-config*

Accediendo a opciones avanzadas se activa la opción VNC.

Una vez realizado esto, el siguiente paso es obtener la dirección IP de la Raspberry Pi. Un dato importante para realizar la conexión es que la Raspberry Pi y el portátil convencional deben estar conectados a la misma red WIFI. Relativo a esto, la dirección IP de la Raspberry no es estática, por lo que, va a variar en función de la red WIFI donde se ha conectado.

La configuración que se realiza en el portátil convencional es simplemente acceder a la página web oficial de VNC [23] y descargar "*VNC Viewer"* dependiendo de las características del portátil, ya sea x64 o x84.

Para hacer que la dirección IP de la Raspberry Pi no varíe en función de la red WIFI conectada lo mejor es configurar la misma para que tenga una dirección IP estática.

Para este proyecto en particular, se optado por establecer una conexión a la nube. Se trata de una conexión vía internet entre la Raspberry Pi y el portátil convencional con una encriptación de extremo a extremo. Para establecer esta conexión, únicamente se requiere la creación de una cuenta en [23] y acceder a la cuenta a través de ambos dispositivos cliente-servidor.

La principal ventaja de esta conexión es que no se requiere conocer la dirección IP de la Raspberry Pi, así como establecer una dirección IP estática, solo es necesario que la placa Raspberry Pi disponga de acceso a internet.

### 4.1.3 Configuración cámara e instalación OpenCV

En este punto, se va a hacer énfasis en los diferentes comandos que nos permiten realizar una correcta configuración del módulo de cámara que dispone la Raspberry Pi. También, se va mostrar los pasos seguidos para la descarga e instalación de la versión 3.0.0 de OpenCV. Se ha optado por la versión 3.0.0 ya que se trata de una versión altamente consolidada frente a otras versiones más recientes donde se desconoce cuál va ser respuesta ante posibles errores. Para realizar estas tares se va a hacer uso en todo momento de la terminal de comandos.

**Configuración Cámara**

Para poner en marcha el módulo de la cámara (RaspiCam) se debe introducir este módulo de manera cuidadosa en el puerto CSI que dispone la Raspberry Pi. Hay que tener especial cuidado con el módulo de la cámara, pues es altamente sensible ante cargas estáticas.

Una vez realizada la conexión física anterior, se accede al menú de configuración de la Raspberry Pi haciendo uso del comando:

- *sudo rasp-config*

La opción 5 permite activar la camera, "*Enable camera*". Esto sería todo. Para comprobar que se ha realizado correctamente, se puede tomar una foto, tras 5 segundos de video, desde la cámara.

- *Raspistill –o prueba.jpg*

Si se desea grabar un video se hace uso del comando:

- *Raspivid –o video.h264 –t 20000*

Donde t indica el tiempo de grabación. Con el anterior comando, el video tiene formato .raw si se desea en otro formato se debe realizar la pertinente conversión.

**Instalación OpenCV**

Para instalar OpenCV se comprueba que se dispone almacenamiento suficiente, es recomendable disponer de 2 Gb a 2,5 Gb.

Primero se realiza una actualización al firmware de la Raspberry Pi:

- *Sudo apt-get update*
- *Sudo apt-get upgrade*

Seguidamente se instala la herramienta de desarrollo, así como las diferentes librerías

- *Sudo apt-get install build-essential cmake-curses-gui pkg-config*
- *Sudo apt-get install libatlas-base-dev gfortran*
- *Sudo apt-get install*\
  *Libjpeg-dev*\
  *Libtiff5-dev*\
  *Libjasper-dev*\
  *Libpng12-dev*\
  *Libavcodec-dev*\
  *Libavformat-dev*\
  *Libswscale-dev*\
  *Libeigen3-dev*\
  *Libxvidcore-dev*\
  *Libx264-dev*\
  *Libgtk2.0-dev*

Descargar OpenCV:

- *Mkdir /home/pi/opencv*
- *Cd /home/pi/opencv*
- *Wget -0 opencv.zip https://github.com/ItSeez/opencv/archive/3.0.0.zip*
- *Wget-0opencv_contrib.zip https://github.com/ItSeez/opencv_contrib/archive/3.0.0.zip*
- *Unzip opencv.zip*
- *Unzip opencv_contrib.zip*

Ejecutar e instalar OpenCV:

Se crea la carpeta build en la carpeta de OpenCV

- *Cd/home/pi/opencv/opencv-3.2.0*
- *Mkdir build*
- *Cd build*
- *Cmake –D CMAKE_BUILD_TYPE=RELEASE*
  *-D CMAKE_INSTALL_PREFIX=/usr/local*
  *-D BUILD_WITH_DEBUG_INFO=OFF*
  *-D BUILD_DOCS=OFF*
  *-D BUILD_EXAMPLES=OFF*
  *-D BUILD_TESTS=OFF*
  *-D BUILD_opencv_ts=OFF*
  *-D BUILD_PERF_TESTS=OFF*
  *-D INSTALL_C_EXAMPLES=ON*
  *-D INSTALL_PTTHON_EXAMPLES*
  *-D OPENCV_EXTRA_MODULES_PATH=.../.../opencv_contrib-3.0.0/modules*
  *-D ENABLE_NEON=ON*
  *-D WITH_LIBV4L=ON*
  *..*
- *Cmake. /*

  Configuramos y generamos. Para ejecutar OpenCV se hace uso del comando:

- *make*

Se procede a realizar la instalación:

- *sudo make install*
- *sudo idconfig*

Un punto a tener en cuenta a la hora de realizar la instalación es poder acceder a la cámara de la Raspberry Pi usando el lenguaje y clases propias de OpenCV. Pues bien, de esta manera, se consigue no realizar un código más enrevesado usando los comandos definidos anteriormente (*Raspistill, Raspivid*)

Por ejemplo, se podrá usar la clase "VideoCapture" del que dispone la librería OpenCV, accediendo a la terminal de la Raspberry y realizando estos pasos:

1-Descargar la librería de Video para Linux

- *Sudo apt-get –y install libv4l-devg v4l-utils*

2- Se carga el módulo de la cámara

- *Sudo modprobe bcm2835-v4l2*

Esta tarea se debe realizar en cada una de las ocasiones en la que se pretenda usar la cámara. Para solucionar esta situación se debe acceder al archivo modules.conf y añadir a la última línea del fichero el nombre del módulo, bcm2835-v4l2.

- Cd /etc/modules-load.d
- Sudo nano modules.conf

## 4.2 CONCEPTOS TEÓRICOS

En este apartado se explican los fundamentos teóricos de los diferentes métodos de detección y seguimiento de objetos que se han tenido en cuenta para la elaboración de un algoritmo de visión robusto y fiable.

En este proyecto, en particular, se van a analizar dos tipos de clasificadores, que son los que se ha considerado que pueden proporcionar una buena detección de los objetos de interés: coches, cara, cuerpo humano completo. El primero será el clasificador basado en una sucesión de clasificadores en cascada y el otro, el clasificador SVM o "Support Vector Machine".

Antes de comenzar, es necesario conocer el concepto de clasificador. Por clasificador, se entiende aquel tipo de sistema que dispone de la capacidad de decidir la pertenencia de un elemento cualquiera a una clase o no, es decir, es un sistema que permite conocer si un elemento tiene relación con una clase o no.

Los dos clasificadores mencionados anteriormente son clasificadores binarios, pues en el caso que se aborda, al clasificador le corresponde la función de asignar una etiqueta al elemento de estudio en función de la similitud con una clase u otra. La comparación de estas clases con el elemento u objeto de estudio se realiza con las características descriptivas que se hallan escogido para el objeto o elemento en cuestión.

Mediante el empleo del clasificador SVM, las características que mejor describen los objetos en una imagen son los descriptores HOG o descriptores basado en histogramas orientados. Para los clasificadores en cascada las características que mejor definen a los objetos son las características Haar o LBP.

A continuación, se procederá a realizar un estudio más exhaustivo de los tres métodos mencionados.

### 4.2.1 Histograma de gradientes orientados (HOG) + SVM

Por descriptor de características se entiende aquel vector compuesto de las características que definen completamente a un objeto. Pues bien, para procesos de detección es imprescindible disponer de una descripción adecuada del objeto que se pretende detectar en una imagen. Hay que recalcar que dependiendo de la aplicación que se aborde el descriptor de características lo formarán unas características u otras. En este sentido, es mejor escoger características que permitan discriminar unas clases de otras y supriman la existencia de diferencias en una clase.

El descriptor HOG o descriptor basado en histogramas de orientación se trata de un descriptor avanzado altamente utilizado por su eficacia. Se trata de un descriptor que cuenta con una alta invariancia a brillos, cambios de iluminación o sombras. También destaca por su detalle en bordes y texturas.

En el año 2005 se presentó un artículo **"Histogram of Oriented Gradients for human detection"** [10], donde se analiza la detección de personas en una imagen a través de histogramas de orientación y que a la postre supuso la base para realizar la detección de cualquier otro tipo de objeto de interés.

Como se ha comentado anteriormente para realizar el proceso de detección primero se deben sacar alguna o varias características del objeto de interés y posteriormente realizar el proceso de clasificación. En este caso, las características se sacarán por medio del descriptor HOG y el proceso de clasificación se realizará por medio de SVM.

**Descriptor HOG (Histograma de gradientes orientados)**

A diferencia de otro tipo de descriptores como puede ser LBP, que se basa en la información de textura que se puede obtener de la imagen, el descriptor HOG se basa en la extracción de información por medio del gradiente de la imagen, mediante el cual se realiza la detección de bordes de objetos presentes en la imagen.

Se entiende como gradiente, G, aplicado sobre una imagen f (x, y) a:

$$\nabla f(x,y) = [G_X \; G_Y] = \left[\frac{\partial f}{\partial x} \; \frac{\partial f}{\partial y}\right]$$

$$|\nabla f| = \sqrt{{G_X}^2 + {G_Y}^2}$$

$$\angle \nabla f = arctan\left({G_Y}/{G_X}\right)$$

El vector gradiente que se muestra en la figura anterior indica la variación máxima para el punto (x, y). También se muestra la definición del módulo y dirección del vector gradiente. Para imágenes digitales discretas el gradiente se aproxima a:

$$\nabla f(x,y) = [G_X \; G_Y] = \left[\frac{\Delta f}{\Delta x} \; \frac{\Delta f}{\Delta y}\right]$$

Y se puede representar por las máscaras:

- $G_x = \frac{\Delta f}{\Delta x}$

| | |
|---|---|
| -1 | 1 |

- $G_y = \frac{\Delta f}{\Delta y}$

| |
|---|
| -1 |
| 1 |

Uno de los principales inconvenientes que muestran las máscaras mostradas en la figura anterior es su alta sensibilidad al ruido. Hay otro tipo de filtros que además de calcular el gradiente, suavizan los pixeles de la imagen y son menos propensos al ruido.

Se va a tomar una imagen original cualquiera y se va a ver el efecto de los diferentes filtros para el cálculo del gradiente.

**Figura 11. Imagen Lena**

- Operador Roberts

| | |
|---|---|
| 0 | −1 |
| 1 | 0 |

| | |
|---|---|
| −1 | 0 |
| 0 | 1 |

**Figura 12. Operador Roberts Eje X**

- Operador Prewitt

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −1 | 0 | 1 |
| −1 | 0 | 1 |

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

**Figura 13. Operador Prewitt Eje X Eje Y**

- Operador Sobel

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −2 | 0 | 2 |
| −1 | 0 | 1 |

| | | |
|---|---|---|
| −1 | −2 | −1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

**Figura 14. Operador Sobel Eje X Eje Y**

Tal como se muestra en todas las figuras anteriores, sea cual sea el filtro usado, el gradiente aporta información de interés para la detección y reconocimiento de objetos. Un valor alto del gradiente se corresponde con el borde o silueta del objeto. El descriptor HOG combina la información obtenida del gradiente en histogramas locales, calculados en celdas uniformes y de reducidas dimensiones, distribuidas a lo largo de toda la imagen.

**Figura 15. Descriptor HOG a imagen con objeto persona [29]**

Para cada una de las celdas en las que se ha dividido la imagen se puede ver la orientacion de los diferentes bordes para cada una de las zonas de la imagen.Esta información extraida de la imagen, permite conocer la forma de los objetos y por ende una buena base para hacer la detección de los mismos.

Para finalizar los histogramas calculados localmente se combinan en bloques de un tamaño mayor.Estos bloques sirven de normalización final de la imagen ademas de conseguir mayor robustez contra cambios de iluminación y distorsiones de la imagen.En este sentido el resultado final es la agrupación de todos los bloques.

## SVM(“Suport Vector Machine”)

Mediante el clasificador SVM es posible encontrar la frontera que permita separar las muestras positivas de las muestras negativas a analizar.Se trata de un clasificador lineal, por lo que en la frontera de decisión en el caso de un espacio de 2 dimensiones será una linea.En el caso de que el número de dimensiones sea mayor, la frontera será un hiperplano.

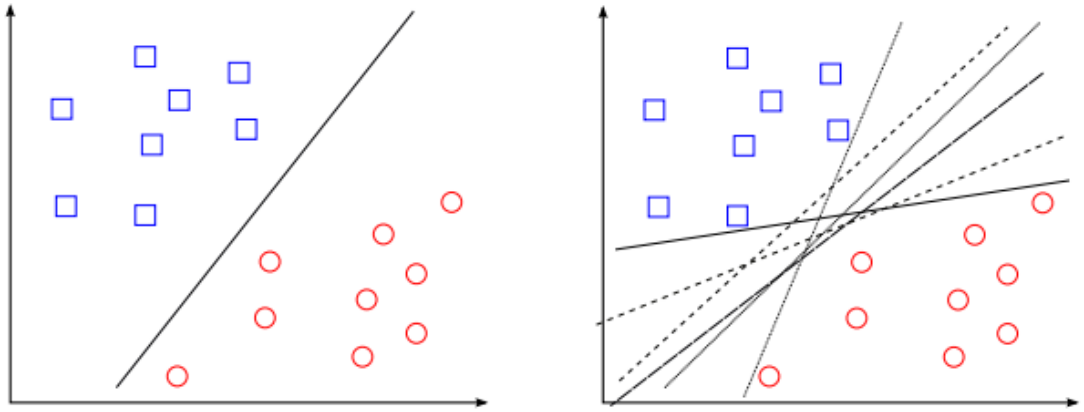

**Figura 16. Frontera de decisión para espacio de 2 o n dimensiones**

Para obtener el hiperplano mencionado se debe maximizar la distancia entre las muestras positivas y negativas mas cercanas empleadas para realizar el entrenamiento, lo que se conoce como, margen máximo.Las muestras que se utizan para maximizar las distancias y calcular el hiperplano se denominan vectores de soporte.

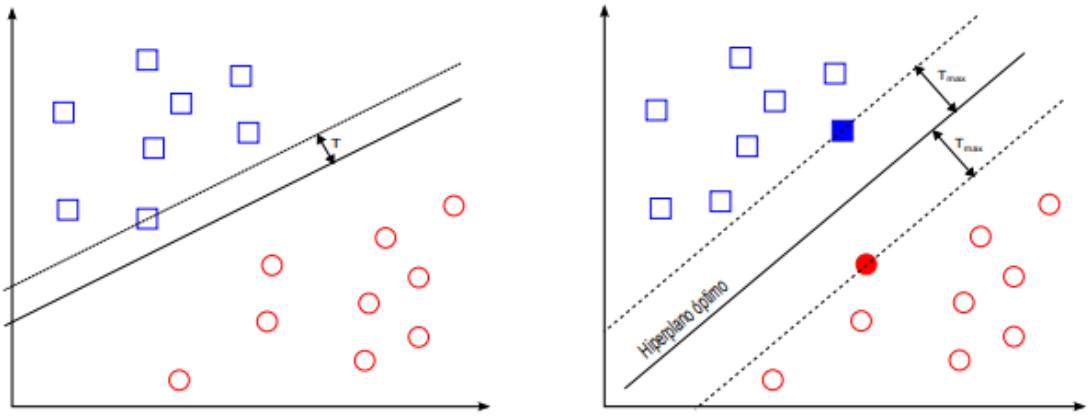


**Figura 17. Margen máximo y vectores de soporte**

En ciertos casos, los datos de entrada son dificilmente separables de forma lineal, por lo que es dificil encontrar un hiperplano.Para conseguir solucionar esta situación SVM proporciona una tecnica conocida como “kernel trick”, mediante la cual se puede transformar el espacio de caracteristicas no separable linealmente en otro espacio separable linealmente sin realizar calculos de manera explícita.

Para finalizar, en la próxima figura, se muestra un esquema de cuales son los diferentes pasos a seguir para extraer las caracteristicas del objeto y su posterior clasificación usando HOG como descriptor de la imagen y SVM como clasificador.

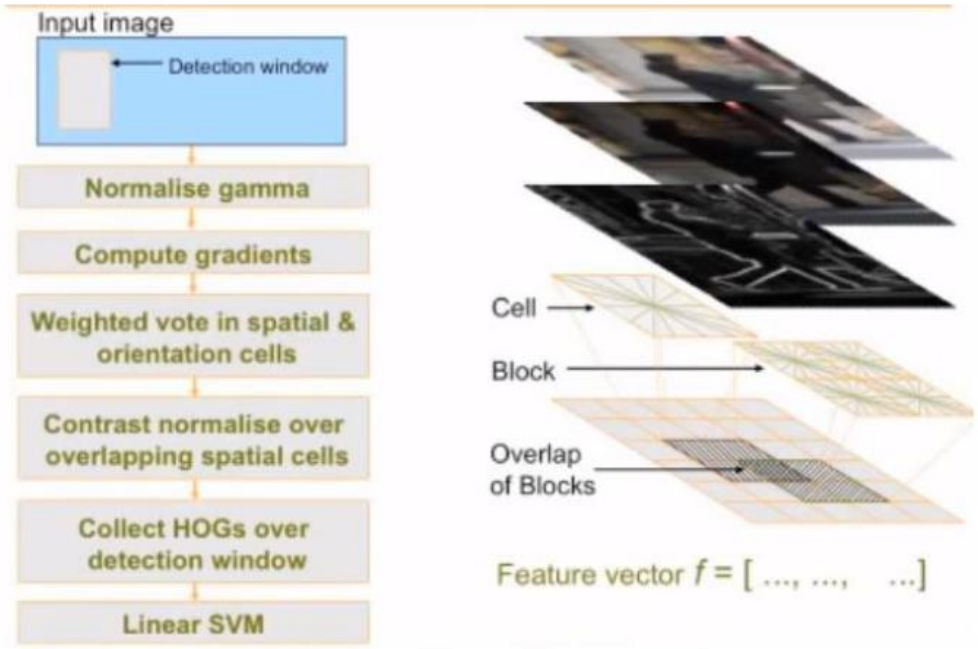


**Figura 18. Esquemas pasos método HOG + SVM [29]**

A continuación se muestra el resultado obtenido a través de la creación de un algoritmo de detección de personas usando histrogramas de gradientes orientados como descriptor y SVM como clasificador.

**Figura 19. Resultado_ Un Objeto Humano [29]**

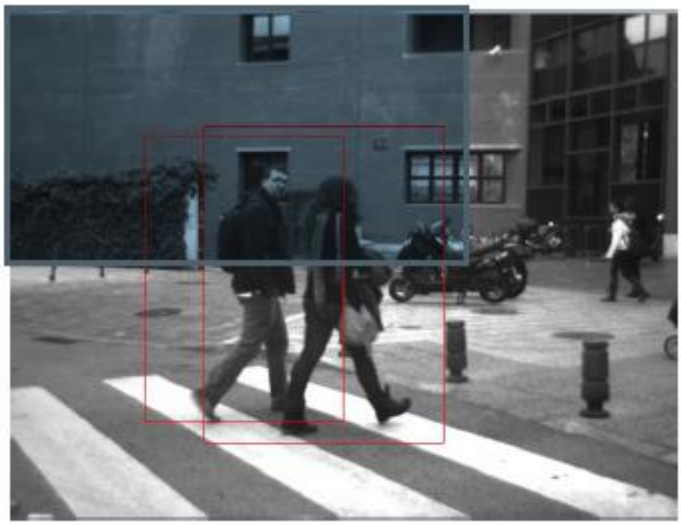

**Figura 20. Resultado_ Varios Objetos Humanos [29]**

### 4.2.2 Descriptor LBP ("Local Binary Patterns")

Otro de los descriptores a analizar es el descriptor LBP ("Local Binary Patterns"). Se trata de un descriptor con unas buenas prestaciones, pues bien, destaca por su invariancia a cambios monotónicos del nivel de gris, así como a cambios de translación.

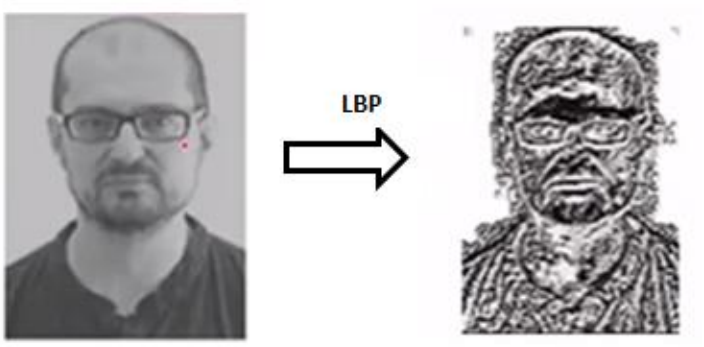


**Figura 21. Transformación LBP**

En la imagen de la izquierda se aprecian unos niveles de gris que va de 0 a 255 y la imagen de la derecha la imagen aplicada una transformación LBP. Para cada uno de los pixeles de la imagen se aplica un código LBP.

Para calcular la transformación a LBP de cada uno de los pixeles, se toma un pixel como pixel de análisis, en este caso se ha escogido el pixel central de la imagen. A continuación, se define una vecindad del pixel e ir comparando cada uno de los pixeles vecinos con el pixel de análisis. En este caso se ha definido como pixeles vecinos a todos los pixeles que tocan al pixel central (los valores 8, 2,7... son los niveles de gris de cada pixel).

| | | | | |
|---|---|---|---|---|
| | **8** | **2** | **7** | |
| | **2** | **7** | **8** | |
| | **7** | **7** | **1** | |
| | | | | |

Ahora se toma un pixel vecino arbitrario, si bien, siempre se debe seguir el mismo orden. Seguidamente se aplica la regla que se muestra:

Valor del bit
- 1 si el valor del vecino es mayor o igual al valor del central
- 0 en caso contrario

Obteniéndose un código binario, que finalmente se pasa a decimal. Este valor se corresponde con el valor del código LBP. Con este código LBP no solo se obtiene el valor del nivel de gris de pixel central, también se obtiene la relación con los pixeles vecinos.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **1** | **0** | **1** | **1** | **0** | **1** | **1** | **0** |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | **182** | | |
| | | | | |
| | | | | |

En los bordes de la imagen no se tienen todos los pixeles vecinos. En este sentido, dependiendo de cuál sea la aplicación, se hacen ciertas suposiciones o en ocasiones directamente no se calculan los LBP para esos bordes o esquinas. El descriptor LBP se define únicamente para imágenes en niveles de gris.

Una vez que se han obtenido las imágenes LBP, lo que se suele hacer es definir un histograma normalizado.

Descriptor LBP

El vector de características se corresponde con cada uno de los códigos LBP asociados a los pixeles de la imagen original. En la imagen el tamaño de la ventana es de 64x128 por lo que la dimensión del vector de características es de 64x128=8192.

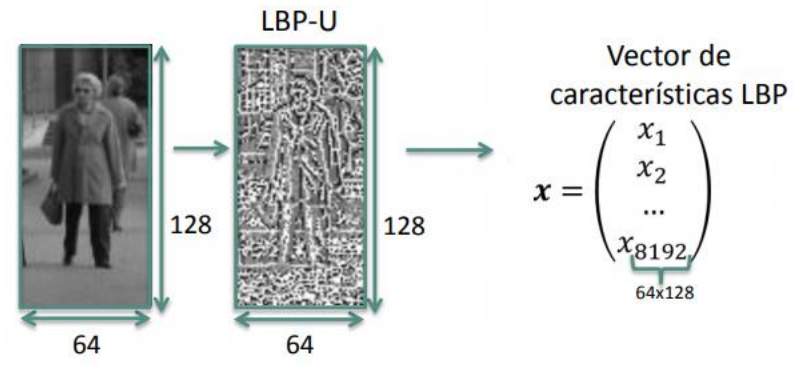


**Figura 22. Vector de características LBP [24]**

Uno de los principales problemas que pueden aparecer con estos descriptores es que dadas dos ventanas con el mismo contenido puede que los códigos LPB, siendo iguales, se encuentren en posiciones totalmente diferentes en la longitud del vector de características o descriptor.

**Figura 23. Inconveniente Vector Características LBP [24]**

Para solucionar este inconveniente se emplean histogramas. El objetivo es conseguir que las ventanas que contengan objetos de interés tengan un histograma muy parecido entre sí, y a su vez, que su histograma sea muy diferente a las ventanas que solo sean fondo.

Otro inconveniente es que al ser los LBP invariantes a la traslación si se corta la venta por la mitad y se transponen ambas subventanas, el histograma inicial y una vez realizadas las operaciones es prácticamente el mismo. Ambos descriptores son muy parecidos. Aparece una mayor cantidad de falsos positivos.

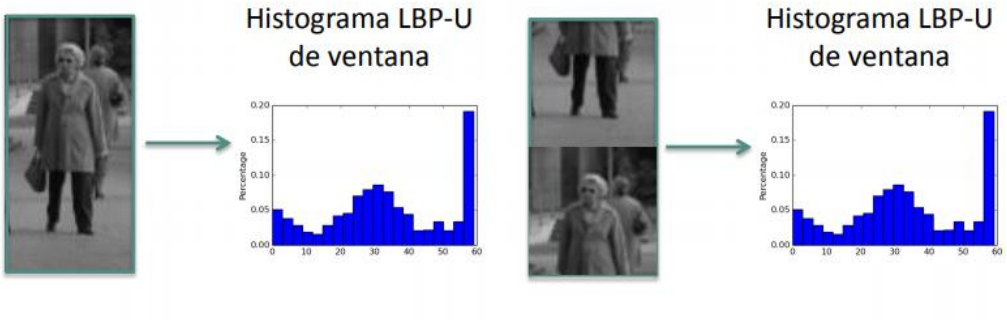


**Figura 24. Inconveniente Histograma LBP [24]**

Para solucionar este inconveniente se divide la ventana en bloques, se hace un histograma para cada uno de los bloques de manera separada. El descriptor de toda la imagen va a ser el resultado de concatenar todos los histogramas obtenidos. De esta manera, se consigue que los histogramas reflejen la diferencia entre ambas imágenes. El descriptor de cada subventana es diferente. Todos los histogramas son normalizados.

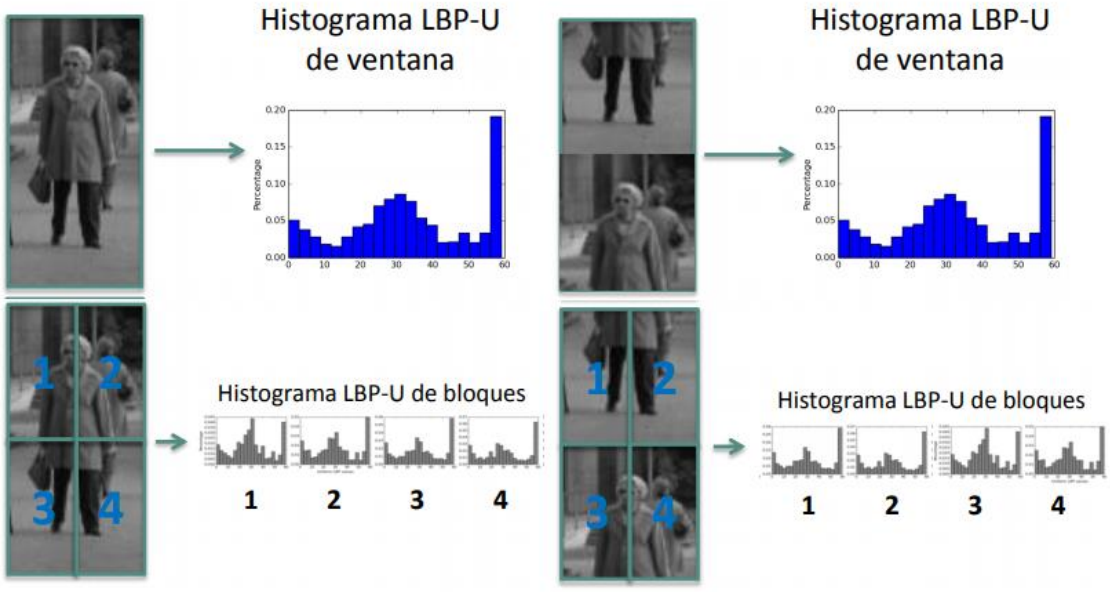


**Figura 25. División y Concatenación de histogramas LBP [24]**

### 4.2.3 Características HAAR + Clasificador en cascada

Los clasificadores en cascada basados en características Haar, como descriptores de la imagen, permiten detectar diferentes objetos de interés presentes en una imagen o en una secuencia de imágenes. Este método de detección fue propuesto por Paul Viola y Michael Jones en su escrito [7]. Se trata de un método basado en aprendizaje automático en donde el algoritmo se entrena en función de un alto número de imágenes positivas y negativas que posteriormente sirven de apoyo para para encontrar los objetos de interés en otras imágenes.

Una de las principales ventajas de los clasificadores en cascada es su bajo coste computacional y su velocidad a la hora de detectar los objetos en imágenes. El modo de operación de este clasificador es procesar la información en escala de grises. Para decidir si en una imagen hay presencia de un objeto de interés, se divide la imagen en varias subregiones de diferentes tamaños y se hace uso de clasificadores en cascada cada uno con una serie de características y cada uno de los clasificadores analiza si la subregión pertenece o no al objeto que se busca. Por medio de este método solo se analizan zonas que se sabe que puede haber presencia de un objeto de interés.

Por lo dicho, como descriptor de la imagen se van a usar las características de Haar. La manera más eficaz de obtener estas características es pasando la ventana de detección a diferentes escalas por toda la imagen. Se estima que pueden extraerse cerca de 100000 características de Haar por imagen.

Por otro lado, el método de clasificación es Adaboost. Se trata de un clasificador muy adecuado para trabajar cuando se dispone de una gran cantidad de características que describen la imagen, permite determinar y seleccionar aquellas características más adecuadas para la clasificación.

Este método de detección se fundamenta en el artículo: “Robust Real-Time Face Detection” de viola y Jones publicado en el año 2004.El artículo está orientado principalmente hacia la detección de caras. Ahora bien, este método se puede emplear para la detección de cualquier otro objeto de interés con alto grado de fiabilidad

En comparación con los descriptores LBP Y HOG, descritos anteriormente, hay que recordar que ambos se van basan en la extracción de características locales de cada pixel, para LBP es la diferencia con sus vecinos y para HOG el gradiente, y donde finalmente, las características obtenidas se aúnan en bloques por medio de histogramas. Para realizar la descripción de la imagen por medio de características Haar se procede a la aplicación de una serie de filtros Haar de diferentes tamaños y en diferentes posiciones de la imagen y cada una de las aplicaciones de este filtro va suponer una componente final del vector de características que describe la imagen.

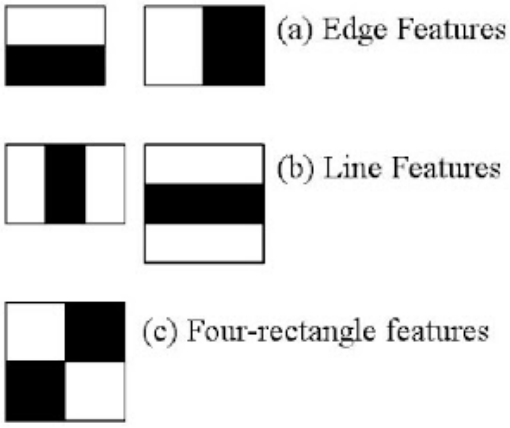


**Figura 26. Filtros Haar [13]**

Los filtros de Haar se aplican por todo el espacio de la imagen. El resultado final de cada filtro es la suma de los pixeles que están en la parte negra menos la suma de los pixeles que están en la parte blanca.

Un filtro de Haar es la combinación de una serie de rectángulos (2 o 4) de iguales dimensiones adyacentes en la dirección horizontal o vertical. Los rectángulos están en negro o en blanco. Los rectángulos de color oscuro dan una contribución positiva al filtro, mientras que los blancos dan una contribución negativa al resultado del filtro.

En este sentido, el resultado del filtro va ser la resta entre la suma de los valores de los pixeles en las zonas negras y las zonas blancas.

| | | | | | |
|---|---|---|---|---|---|
| **250** | **250** | **150** | **150** | **100** | **100** |
| **150** | **150** | **75** | **75** | **75** | **75** |
| **255** | **255** | **255** | **100** | **100** | **100** |
| **200** | **200** | **150** | **150** | **150** | **150** |
| **100** | **100** | **100** | **255** | **255** | **255** |
| **100** | **100** | **100** | **255** | **255** | **255** |

**Tabla 2. Niveles de Gris Imagen_ Filtro Haar**

Tomando una imagen cualquiera el resultado del filtro para ejemplo práctico anterior será:

$$R = \sum_{(x,y)\in Oscuro} I(x,y) - \sum_{(x,y)\in Claro} I(x,y) = 800 - 450 = 350$$

En ciertos casos, se hace necesario normalizar el valor al tamaño de la ventana para conseguir que los valores no cambien al cambiar el tamaño del objeto o que características aplicadas a varias escalas tengan valores cercanos. También puede ser necesaria la normalización para evitar los posibles cambios de iluminación.

Cada uno de los filtros se analiza en diferentes posiciones y orientaciones de la imagen para detectar cambios de intensidad ante estos estas escalas y orientaciones.

Aplicando los filtros Haar con dos o tres rectángulos, el resultado es muy parecido al cálculo del gradiente en la dirección horizontal y vertical. Se centra principalmente en la detección de la silueta o bordes de la imagen. Esto es lógico, pues como se ha explicado

anteriormente, el filtro Haar se basa en la diferencia de intensidades entre pixeles vecinos en las direcciones vertical y horizontal al igual que sucede con el operador gradiente. El filtro con cuatro rectángulos aborda zonas que contienen contornos en la dirección diagonal (45 grados).

**Figura 27. Filtro Haar Cara Humana [25]**

Para conseguir aumentar el rendimiento del método se puede usar filtros de Haar extendidos. Estos filtros suponen añadir nuevas configuraciones a los filtros: rotar, aumentar o extender los filtros básicos para obtener otro tipo de características adicionales.

Como se ha comentado, en este método de descripción, el número de características es muy amplio puesto que un mismo filtro se va a aplicar en diferentes puntos de la imagen y a diferentes escalas. De aquí, se puede intuir que el procesamiento de la imagen puede ser muy costoso.

Para poder hacer frente a este número tan elevado de características que son extraídas por los filtros de Haar se proponen tres alternativas:

1. **Imagen integral**

El objetivo es transformar la imagen original en una imagen integral. En esta imagen, cada uno de sus pixeles va ser la suma de intensidad de los pixeles colocados en la parte superior izquierda para el mismo punto de la imagen original. Esta transformación permite reducir el número de operaciones necesarias para calcular las características de Haar. Como se ha comentado anteriormente, el cálculo de características de Haar se basa en la diferencia de valores de pixeles de diferentes rectángulos, con la imagen integral esta operación se realiza de manera simple por lo que se reduce el tiempo de computación de manera considerable.

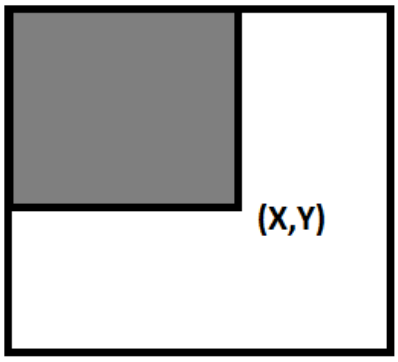


**Figura 28. Imagen Integral**

$$II(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} I(x',y')\,\theta$$

2. **Clasificación basada en Adaboost**

Se trata de la combinación de una gran cantidad de clasificadores simples que se pueden calcular de manera relativamente sencilla para conseguir un clasificador más fuerte. Los clasificadores simples se basan en una única característica de Haar, de manera que en el camino hacia la obtención de un clasificador más potente se pueden ir seleccionando las características que más se adapten a las necesidades, consiguiendo un clasificador de alto rendimiento.

Por otro lado, para cada clasificador simple nuevo, se va hacer más énfasis en aquellos ejemplos en los no se han clasificado de forma correcta en etapas previas, aumentando su peso y reduciendo el peso de los que estén bien clasificados. De esta manera se consigue mejorar el clasificador final para ejemplos con mayor nivel de dificultad.

El clasificador simple o débil se corresponde con un árbol de decisión de profundidad 1, "decisión stump". Este clasificador simple depende de una única característica de Haar.

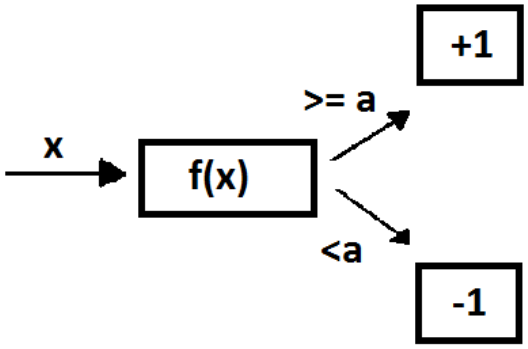


**Figura 29. Clasificador débil_ Árbol de decisión profundidad 1**

Como se puede observar en la imagen, la función f(x) va a ser el resultado de evaluar una característica de Haar en una posición y orientación determinada. La 'a' es un rango que posibilita distinguir ejemplos positivos de negativos. Ahora bien, no se conoce con exactitud si los valores positivos se encuentran antes o después del umbral o igual con los negativos ahí entra en juego otros parámetros.

En definitiva, el método de clasificación Adaboost se basa en el aprendizaje de un determinado conjunto de clasificadores débiles. A medida que se va aprendiendo de los clasificadores débiles (basados en una característica HAAR) se va corrigiendo el peso de cada muestra para la siguiente interacción en función del error del clasificador. La combinación de estos clasificadores débiles, con una amplia variabilidad entre ellos, permite la constitución de un clasificador fuerte que dispone de un alto rendimiento, así como capacidad de generalizar y reducir los errores de clasificación de los objetos de interés.

Por lo dicho, se define como clasificador fuerte a la suma de todos los clasificadores débiles.

### 3. Cascada de clasificadores

Combinación secuencial de todos los clasificadores fuertes que se han obtenido. A partir de una imagen inicial se analiza la presencia de un objeto de interés en la misma. En caso negativo, la imagen se desecha definitivamente. En el caso de que se encuentre un objeto de interés se avanza al siguiente clasificador. Se realiza este procedimiento hasta llegar al final de todos los clasificadores. Solo hay un objeto de interés en la imagen analizada cuando todos los clasificadores detecten la presencia de ese objeto.

Una de las ventajas de este método es que permite reducir el tiempo de cómputo de manera considerable, pues se anula el análisis de las imágenes que no tienen los objetos que se busca a las primeras de cambio o inicio de la cascada de clasificadores.

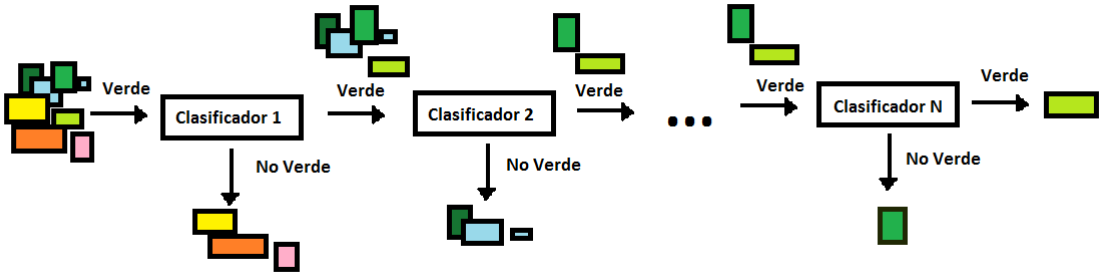


**Figura 30. Cascada de Clasificadores**

La cascada de clasificadores es una forma por la cual se consigue aumentar de manera notoria el proceso de detección de objetos. El objetivo que se pretende alcanzar es rechazar el mayor número posible de imágenes negativas en las primeras etapas de clasificación usando clasificadores que operen con las mínimas características posibles y de ahí usar la mayor cantidad de características en un reducido número de imágenes que tengan mayor dificultad para ser clasificadas.

### Implementación Práctica

Hasta el momento se ha comprobado que a partir de las características de Haar extraídas de las imágenes en combinación con una cascada de clasificadores entrenados cada uno de ellos por medio de Adaboost permite la obtención de un método altamente eficiente y robusto para la detección de un determinado objeto de interés en una subregión de la imagen. Ahora, el objetivo es poder detectar cualquier objeto y a cualquier escala en la imagen.

Previo a realizar todo lo mencionado es conveniente establecer las condiciones del pre-procesamiento de imágenes que se van a usar para realizar el entrenamiento. Para ello se van a abordar dos condiciones principales:

1. Normalización de las dimensiones de la imagen

Normalización de todas las imágenes a un tamaño fijo de 24x24 pixeles para reducir el efecto de las variaciones en las imágenes. A partir de este tamaño se tiene la posibilidad de detectar objetos de mayor escala.

2. Normalización de imágenes para reducir efecto de la iluminación

También es necesaria la normalización de las imágenes para reducir el efecto de la iluminación.

Una imagen con distinta iluminación puede tener diferentes valores para las características de Haar. Para normalizar, lo que se hace es, restar la media de las intensidades de la imagen a la intensidad del punto X, Y, todo ello, divido por la desviación estándar. Llegando a obtener imágenes de media igual a cero y desviación estándar igual a 1.

$$I'(x,y) = \frac{I(x,y) - \mu(I)}{\sigma(I)}$$

### 4.2.4 Seguimiento de objetos ("Tracking")

Para aumentar la capacidad de detección de los diferentes objetos de interés se requiere implementar un algoritmo de seguimiento o ***"tracking"***. OpenCV cuenta con una extensa lista de algoritmos de seguimiento, para este proyecto en particular, se ha considerado que los métodos que mejor encajan son:

1. **Método Lucas-Kanade**

Este método usa flujo óptico para el seguimiento de objetos. Por flujo óptico se entiende al patrón de movimiento que sigue un objeto determinado entre 2 "*frames"* consecutivas a causa del movimiento propio del objeto como de la cámara que adquiere las imágenes.

Se puede emplear flujo óptico para conocer la estructura del movimiento, estabilización de video o entender el movimiento del objeto. Para usar este método se asume que:

- La intensidad de gris de los pixeles de un objeto no cambia de un "*frame"* a otro
- Pixeles de un mismo objeto tiene el mismo movimiento

OpenCV cuenta con la función ***calOpticalFlowPyrLK ()*** que permite el seguimiento de puntos característicos en un video.

Uno de los principales inconvenientes de este método es que necesita un procesamiento elevado de información y con las limitaciones que se tiene para este proyecto, no se ha considerarlo como método ideal de seguimiento.

2. **Detección Blob**

Por "blob" se entiende a un conjunto de pixeles que comparten las mismas características. Una vez se han filtrado e identificado los pixeles conectados que se pretende seguir, se va analizando cada "*frame*" del video la ubicación de pixeles con estas mismas características.

Cotejando bibliografía relacionada con este método se pude concluir que este método no está diseñado específicamente para encontrar objetos específicos en una imagen y es un método que da mejores resultados para seguimiento de formas geométricas con un determinado color.

## 3. Seguimiento por Color. MeanShift

Se trata de un algoritmo de seguimiento de objetos basado en color. Meanshift busca encontrar la zona con mayor densidad de pixeles en una imagen a través del análisis de los histogramas de color y gradientes de densidad

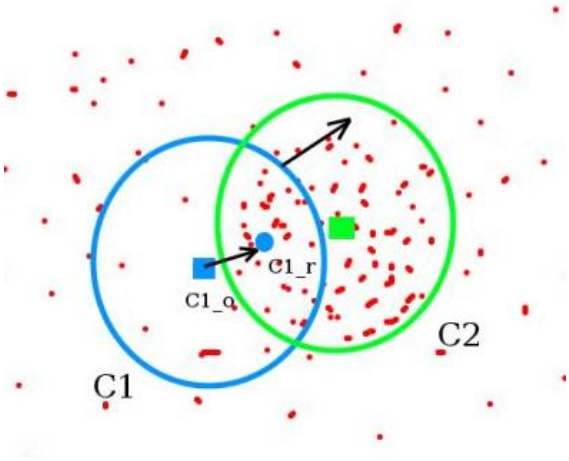


**Figura 31. MeanShift [26]**

El concepto de Meanshift es muy intuitivo. Imagínese un conjunto de puntos distribuidos de manera aleatoria, tomando una ventana cualquiera, el método permite mover la ventana al área de mayor densidad de pixeles.

Este método muestra debilidad cuando un determinado objeto que se pretende detectar se acerca demasiado a la cámara, pues bien, la ventana que se utiliza tiene siempre las mismas características. Para solventar este inconveniente y mejorar la detección existe el método de CamShift.

## 4. CamShift

Como se ha comentado anteriormente la ventana que se utiliza con el método Meanshift es siempre el mismo ya sea que el objeto a detectar se encuentre muy lejos o muy cerca de la cámara. Con este nuevo método se busca adaptar el tamaño de la ventana al tamaño y orientación del objeto a detectar.

Para este método, primero se aplica meanshift y una vez converge se actualiza el tamaño de la ventana en función:

$$S = 2 \text{ x } \sqrt{\frac{M_{00}}{255}}$$

Con este también se consigue calcular la orientación más ajustada para la eclipse que contiene al objeto a detectar.

Una vez ajustado el tamaño de la ventana, y conociendo su localización, se calcula nuevamente MeanShift. Este proceso se repite tantas veces como precisión se desee obtener.

### 4.2.5 Sistema detección Final

A pesar de que existen una gran cantidad de métodos y herramientas para la detección de diferentes objetos que puede haber en una imagen, para este proyecto se ha considerado que los métodos que pueden dar mejor resultado son los basados en extracción de características tipo Haar, características tipo LBP o mediante el descriptor HOG empleando clasificadores en cascada o clasificadores SVM.

Las imágenes se adquieren por medio de una cámara monocular (RaspiCam) colocada de manera estratégica en el dron. La colocación de la cámara supone un paso de gran importancia. Como posibles opciones se puede barajear la colocación de la cámara de tal manera que el eje vertical que pasa por el centro de gravedad del dron pase por el centro de masas de la cámara de manera que se obtengan imágenes de vista cenital. Otra opción es colocar la cámara inclinada 45 grados con respecto al eje anterior.

La primera propuesta de colocación de la cámara tiene un inconveniente, y es que, solo se dispone de una única vista del objeto para poder realizar la detección del mismo, únicamente la vista cenital. En este sentido, al disponer de una única vista del objeto, puede que la detección no se realice de la manera deseada ya que pueden aparecer solapamientos con otro tipo de objetos, de manera que se espera la aparición de una gran cantidad de falsos negativos.

Mediante la colocación de la cámara con cierta inclinación se permite visualizar la presencia de los objetos de interés antes de estar justo encima del mismo. Así también, se pueden tener diferentes ángulos de visualización dependiendo de la dirección de entrada del dron al objeto.

Analizando ambas propuestas, se ha decido suponer como método de adquisición de imágenes la cámara con cierta inclinación.

Por otro lado, otro aspecto de vital importancia en la adquisición de imágenes es conseguir la mayor estabilidad posible del dron a partir del cual se adquieren imágenes. En este proyecto se dispone de un dron con 6 motores de propulsión por lo que se consigue amentar la estabilidad del mismo en detrimento de otras características técnicas como puede ser la velocidad o altura de crucero.

En cuanto a la estrategia de detección a utilizar para detectar objetos de interés presentes en una imagen, se puede decir que los tres métodos descritos anteriormente pueden dar unos resultados aceptables. Ahora bien, analizados todos los casos se puede afirmar que el mejor método para abordar este proyecto es utilizando características Haar como descriptor y clasificadores en cascada como método de clasificación.

Se trata de un método que destaca por su precisión y rapidez. En muchos artículos y libros se ha demostrado la gran precisión que puede obtener para la detección de rostros humanos, aquí se pretende analizar si es posible obtener resultados a la altura para diferentes objetos presentes en la imagen. Otro punto a destacar de este método es que la creación del clasificador se hace antes de la ejecución del programa y que una vez realizado el entrenamiento el archivo .xml ocupa muy poca memoria. Esto, permite que el tiempo de procesamiento de imágenes pueda realizarse prácticamente en tiempo real.

En apartados siguientes se explicará los pasos a seguir para la creación de un detector de objetos fiable y optimizado basado en características Haar.

Por otra parte, métodos como HOG +SVM requieren sistemas con mayor capacidad de computo, por lo que haciendo uso únicamente de la Raspberry Pi (características limitadas) los resultados obtenidos para realizar una detección en tiempo real son relativamente pobres, dada la lentitud para procesar gran cantidad de imágenes.

### 4.3 Entrenamiento Clasificadores HAAR

Por defecto, OpenCV cuenta con una serie de clasificadores ya entrenados, como puede ser el clasificador de cara, ojos o cuerpo completo. Ahora bien, existen objetos que no existen por defecto por lo que es necesario realizar una etapa de entrenamiento.

En este punto se va mostrar cuales son los pasos a seguir para entrenar a un ordenador a que realice la tarea de reconocimiento de cualquier tipo de objetos de interés. En este caso, para entrenar una Cascada Haar se requiere disponer de dos tipos de muestras: positivas y negativas. Cuanto mayor sea el número de muestras mejor será el resultado final.

Cuando se habla de muestras positivas, se hace referencia a imágenes en las que se encuentran los diferentes objetos de interés que se desee detectar. Por lo contrario, las muestras negativas, son imágenes en las que no debe aparecer ningún objeto de los que se pretenda detectar. Para entender mejor el proceso a seguir para realizar el entrenamiento se va a dividir el proceso en diferentes pasos:

1. Búsqueda y recolecta de imágenes.

   Se requiere recolectar una muestra de imágenes positivas y otra muestra de imágenes negativas. Es preferible y aconsejable recolectar más imágenes positivas que negativas.

2. Creación de ficheros de ubicación

   Mediante el empleo de un editor de texto, se van a crear archivos de texto con extensión .info o. idx. Se van a realizar 2 archivos, uno para las muestras positivas y otro para las muestras negativas.

   La estructura del archivo para las muestras positivas ha de ser como sigue:

   <Ruta>/img_nombre nobj x1 y1 w1 h1 x2 y2…

   Cada una de las imágenes positivas que se va a emplear para este entrenamiento debe disponer de una mención en el archivo. idx

   Con ruta se pretende expresar la ubicación de memoria en donde se encuentra la imagen. ‘Img_nombre’ hace referencia al nombre de la imagen en cuestión.

   Con ‘nobj’ se indica la cantidad de objetos de interés presentes en la imagen. Las coordenadas ‘x1’ ‘y’1,’ x2’ ‘y2’ indican la ubicación de los objetos de interés en la imagen. Por último, los parámetros ‘w’ ‘h’ reflejan las dimensiones del rectángulo que contiene al objeto de interés.

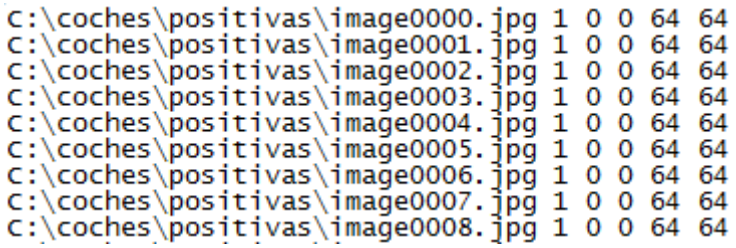


**Figura 32. Fichero Muestras Positivas**

En la figura anterior se muestra una captura del fichero y la correspondiente estructura para realizar la fase de entrenamiento para la detección de coches. Para la elaboración de esta fase de entrenamiento se han cogido como imágenes de muestra positiva cerca de 2000 imágenes, y, por otro lado, unas 1950 imágenes como muestra negativa.

Las imágenes de muestra se encuentran recortadas para que solo contengan al objeto de interés, coches en este caso. Es por ello por lo que en la figura anterior se aprecia que todas las imágenes son del mismo tamaño 64x64 y por lo que se ha comentado el objeto va abarcar todas las dimensiones de la matriz. Del mismo modo solo existe un objeto de interés en cada imagen y las coordenadas 'x' 'y' hacen referencia en este caso al origen de coordenadas de la imagen.

Tal como se ha comentado, también es necesario crear un archivo .info o .idx para conocer la ubicación y características de las imágenes de muestra negativa. El procedimiento es el mismo que para la elaboración del fichero para muestras positivas salvo que para este caso no es necesario especificar las características de los objetos de interés presentes en la imagen, ya que en estas muestras no hay presencia de objetos de interés.


**Figura 33. Fichero Muestras Negativas**

3. Creación de fichero Vector

Este fichero se corresponde con las imágenes de muestra positiva. Se trata de un fichero binario al que se le va dar el nombre cars.vec para el caso de conseguir detectar coches como objeto de interés.

La librería de OpenCV incluye por defecto una aplicación de nombre "*opencv_createsamples"* con la que es posible crear el fichero vector mencionado. Los pasos que sigue la aplicación para lograr el archivo. vec es tomar cada una de las imágenes positivas, las normaliza y las reescala al tamaño deseado finalmente se genera el archivo. vec

Accediendo a la terminal de comandos del sistema operativo para generar el fichero binario. vec se escriben estos comandos:

- *Cd C:\opencv\build\x64\vc11\bin*
- *Opencv_createsamples –vec cars.vec -info cars.info -w 20 –h 20*

Los parámetros de ‘w’ y ‘h’ muestran el alto y ancho del reescaldo, ‘cars.info’ el fichero de ubicación de las imágenes y ‘cars.vec’ el fichero binario de salida.

4. Cargar Clasificador en Cascada

Una vez se tiene el archivo binario. vec se procede a crear el clasificador en cascada con extensión .xml. Para realizar esta tarea OpenCV cuenta con la aplicación que viene por defecto “*opencv_traincascade”*. La aplicación toma las muestras positivas de tal manera que se ha especificado en el archivo. vec y las muestras negativas de forma aleatoria.

En este caso el comando empleado para realizar esta labor es:

- *opencv_traincascade -data C:\coches -vec cars.vec -bg neg.info -numPos 2000 -numNeg 1950 -numStages 12 -featureType HAAR -minHitRate 0.999 -maxFalseAlarmRate 0.5 -w 20 -h 20*

En este comando se aprecia que la ubicación del archivo .xml será ‘C:\coches’. Se han añadido los pertinentes archivos tanto el archivo. vec para las muestras positivas como el archivo .info de las muestras negativas. También se ha establecido el tipo de cascada como tipo Haar y el número de muestras positivas como negativas. ‘NumStages’ indica el número de etapas que va a disponer el clasificador, en este caso, 12. El mínimo valor de acierto viene definido por ‘minHitRate’ y por su parte ‘maxFalseAlarm’ hace referencia al máximo valor de fallo.

Por último, los parámetros de altura y anchura ‘w’, ‘h’ han de ser lo mismo que los definidos anteriormente para la elaboración del fichero. vec.

Como es lógico pensar, es de libre elección poner los valores que se desee los diferentes tipos de parámetros. Ahora bien, hay que tener presente que en función de los diferentes valores de los parámetros y del número de muestras utilizado, se hace influencia en el tiempo de computo necesario para completar el proceso de entrenamiento. Ligado a esto se puede establecer una relación simple por la que se puede afirmar que cuanto mayor sea el tiempo de cómputo en la etapa de entrenamiento mayor será la precisión resultante de ese entrenamiento.

Una vez realizado todo el proceso anterior se puede comprobar la existencia de un archivo .xml en la ruta especificada.

## 4.4 Resultados Visión Alcanzados

Una vez explicados los apartados teóricos que se van a tener en cuenta para la elaboración del proyecto, se va a proceder a realizar la implementación de los diferentes algoritmos para la detección de los diferentes objetos de interés.

La Raspberry Pi, como se ha comentado en apartados anteriores de esta memoria, dispone de una capacidad de cómputo baja por sus características técnicas. Para mejorar y agilizar el diseño del algoritmo se van a realizar las diferentes etapas de entrenamiento en un ordenador portátil convencional hp ProBook 640 G1 con las siguientes características:

- Windows 7
- Ubuntu 16.04
- 8 GB de RAM
- Procesador Intel Celeron a 2 GHz

Se trata de un ordenador de gama media, pero que, por sus características técnicas, puede mejorar y agilizar los diferentes procesos de entrenamiento. En este sentido, el procedimiento de trabajo que se va a seguir es diseñar los algoritmos de detección, así como las etapas de entrenamiento en la plataforma Windows o Ubuntu, se realizaran los pertinentes cambios y comprobaciones y finalmente se pasará el algoritmo completamente funcional a la plataforma de Raspberry y se comprobará el correcto funcionamiento del algoritmo de detección escogido.

### 4.4.1 Lenguaje de Programación: C++

Para llevar a cabo el diseño del algoritmo de visión usando C++ como lenguaje de programación se va a descargar e instalar la versión 3.0.0 de librería OpenCV para Windows. El entorno de programación escogido ha sido la versión más reciente de Microsoft Visual Studio. Este entorno de desarrollo es compatible con una gran cantidad de lenguajes de programación ente los que se encuentra el lenguaje de programación C++.

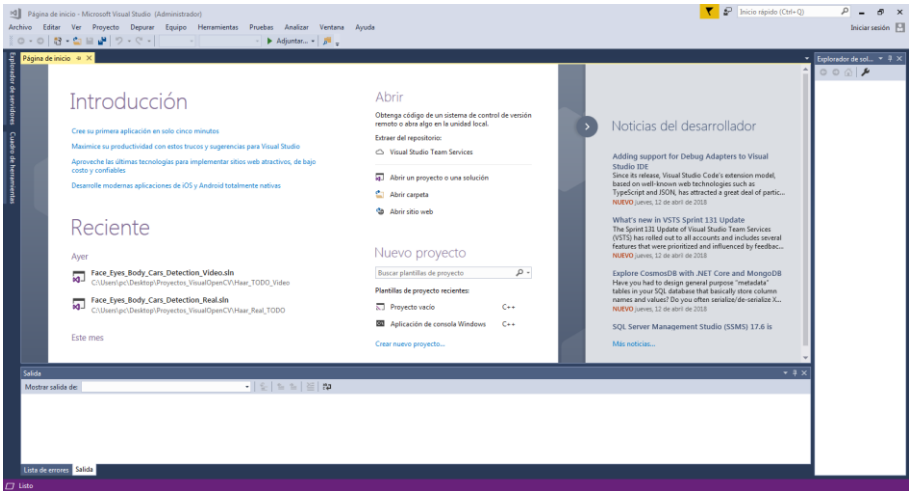


**Figura 34. Interfaz Microsoft Visual Studio 2017.**

Llegado este punto hay que decir que se han llevado a cabo 2 procesos de entrenamiento, uno para coches y otro para personas. Las fotos usadas para el entrenamiento son fotos aéreas tomadas de una amplia base de datos de acceso público. En ambos casos se ha filtrado la base de datos para dejar únicamente 2000 muestras positivas de cada objeto y 1950 muestras negativas. Para el caso de detección de cara, se ha considerado que el clasificador del que dispone OpenCV cuenta con un índice de aceptación favorable para la parte en la que se va a usar en este proyecto (comprobar el funcionamiento del algoritmo en tiempo real), por lo que no se ha realizado un proceso de entrenamiento para caras.

Una vez realizado el entrenamiento de los clasificadores en cascada, la manera de abordar la detección de un objeto cualquiera es relativamente sencilla. Enseguida se muestra un diagrama de flujo del programa prototipo para realizar la detección.

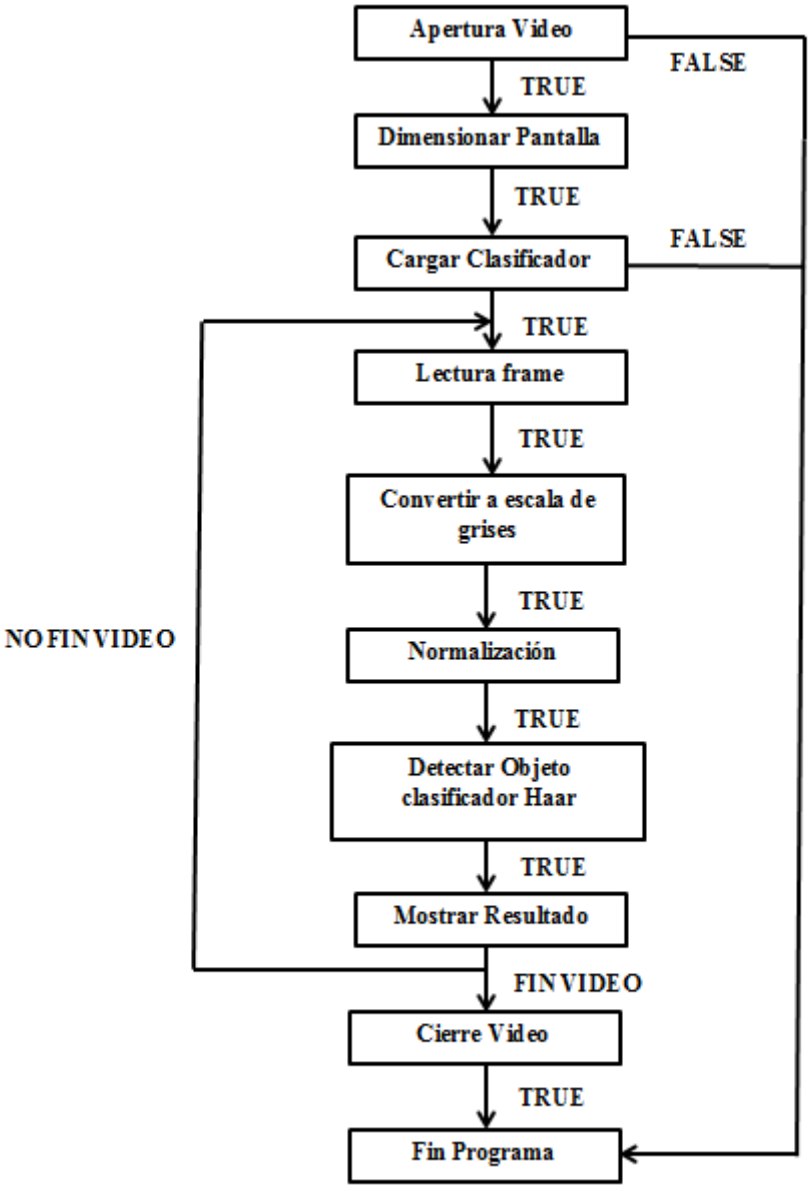


**Figura 35. Flujo de Programa Detección Objetos de Interés**

Acorde a lo planteado, una vez se haya cargado el video o se toma el video real, se hace un reescalado de área del video a analizar. Más tarde, se verá cómo influye el tamaño del video en la detección de objetos. Seguidamente se carga el clasificador Haar, ya entrenado, en caso de que no se haya cargado correctamente se finaliza el programa, pues, no tiene sentido seguir el avance del mismo.

Cuando el video y el clasificador se abran correctamente, se procede a analiza cada una de las capturas del video de forma individual. Se convierte cada "*frame*" a escala de grises lo que permite mejorar el rendimiento del algoritmo, así como, reducir el tiempo de análisis de cada una de las imágenes. Para reducir el efecto de la iluminación se ha considerado la normalización del histograma de la misma, buscando que los valores de brillo y contraste alcancen valores constantes.

Finalmente, se analiza la imagen a través del clasificador HAAR, se muestran por pantalla la detección realizada, se cierra el video y finaliza el programa. OpenCV dispone de una serie de clases y métodos que permiten realizar los pasos comentados de una manera muy cómoda.

- *void cvtColor (InputArray **src**, OutputArray **dst**, int **code**, int **dstCn**=0): C++*

La función "***cvtColor"*** permite convertir una imagen de un determinado espacio de colores a otro. Se pasa como parámetros la imagen o matriz de origen, la matriz de destino y el código de conversión para cambiar de espacio de colores. También es posible especificar el número de canales para la imagen de destino.

Para pasar una imagen a color (RGB) a escala de grises (GRAY) se lleva a cabo la siguiente transformación matemática.

RGB [A] a Gray: **Y←0.299·R + 0.587·G + 0.114·B**

La función "***equalizeHist"*** permite normalizar el brillo e incrementa el contraste de una imagen. Se pasa como parámetros de la función, la imagen inicial que se pretende ecualizar y la imagen de destino con el mismo tipo y tamaño que la imagen inicial.

- *void equalizeHist (InputArray **sr**, OutputArray **ds** ): C++*

El método ***"detectMultiScale"*** de la clase ***"CascadeClassifier"*** permite la detección de objetos de diferente tamaño en la imagen de entrada. Este método devuelve como lista de rectángulos los objetos detectados. Los parámetros que recibe este método son la imagen de entrada, donde se hace la detección de objetos, la lista donde se almacenan los objetos detectados, un factor de reescalado, el mínimo de vecinos para cada candidato a objeto, mínimo y máximo tamaño del objeto a detectar.

- *void CascadeClassifier: detectMultiscale (InputArray **Img**, vector<Rect> &**obj**, double **scaleFactor**=1.1, int **minNeighbors**=3, int flags=0, Size **minSize**= Size(), Size **maxSize**= Size()):* C++

Una vez analizado el algoritmo usado se van a presentar los resultados obtenidos. En las próximas imágenes se muestra el resultado obtenido mediante el diseño y entrenamiento de un clasificador Haar usando 300 muestras positivas y 290 muestras negativas. El tamaño escogido para la ventana 320x240.


**Figura 36. Resultado 1_CocheHaar 300P_ 290N**


**Figura 37. Resultado 2_ CocheHaar 300P_ 290N**

En las dos imágenes mostradas se ve como el algoritmo muestra síntomas de mejora. Se aprecia gran cantidad de falsos positivos, pues bien, se detectan objeto algo que no se corresponde con el objeto a buscar. También aparecen situaciones en las que no se detecta ningún o solo alguno de los objetos de interés presentes en la imagen, esto es, la tasa de verdaderos positivos es muy pobre.

Ampliando la base de datos para el entrenamiento del clasificador HAAR a 2000 muestras positivas y 1950 muestras negativas se aprecian una mejora notable acerca del algoritmo planteado.


**Figura 38.Resultado 1_ CocheHaar 2000P_ 1950N**

Mirando los resultados obtenidos, se puede ver como el algoritmo dispone de mayor capacidad de detección de los objetos presentes. En la imagen se puede notar que solo son detectados los coches u objetos que se encuentran a cierta distancia de la cámara o lo que es lo mismo los objetos o rectángulos que se encuentran en cierto umbral de tamaño por diseño.

En la siguiente secuencia de imágenes se puede ver como cada vez que un vehículo llega a la distancia o tamaño de objeto definido se produce la detección del mismo.


**Figura 39. Resultado 2_ CocheHaar 2000P_ 1950N**


**Figura 40. Resultado 3_ CocheHaar 2000P_ 1950N**

### 4.4.2 Lenguaje de programación Python

Para poder utilizar lenguaje Python se va a correr Ubuntu 16.04 como sistema operativo. La descarga de Python y OpenCV se puede llevar a cabo a través de los siguientes comandos:

- *Sudo apt-get update*
- *Sudo apt-get upgrade*
- *Sudo apt-get install build-essential*
- *Sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev*
- *Sudo apt-get install python-dev python-numpy libtbb2 libjpeg-dev libpng-def libtiff-dev libjasper-dev libdc 1394-22-dev*
- *Sudo apt-get install python-opencv*
- *Sudo apt-get install python-matplotlib*

Para poder hace uso de funciones básicas de procesamiento de imágenes, ya sea, rotaciones, translaciones, detección de bordes o contornos, con Python y OpenCV, se instala la librería *imutils:*

- *Pip install imutils*

De la misma manera que para el caso anterior, se sigue el mismo flujo de programa para la detección de objetos. En el caso de analizar la detección personas se puede ver que en el caso de utilizar el clasificar en cascada tipo Haar los resultados son mejorables. Tomando como medio de análisis la grabación de personas transitando por una vía publica a partir de una cámara fija.

Tomando como clasificador Haar que dispone OpenCV por defecto se obtienen los siguientes resultados:

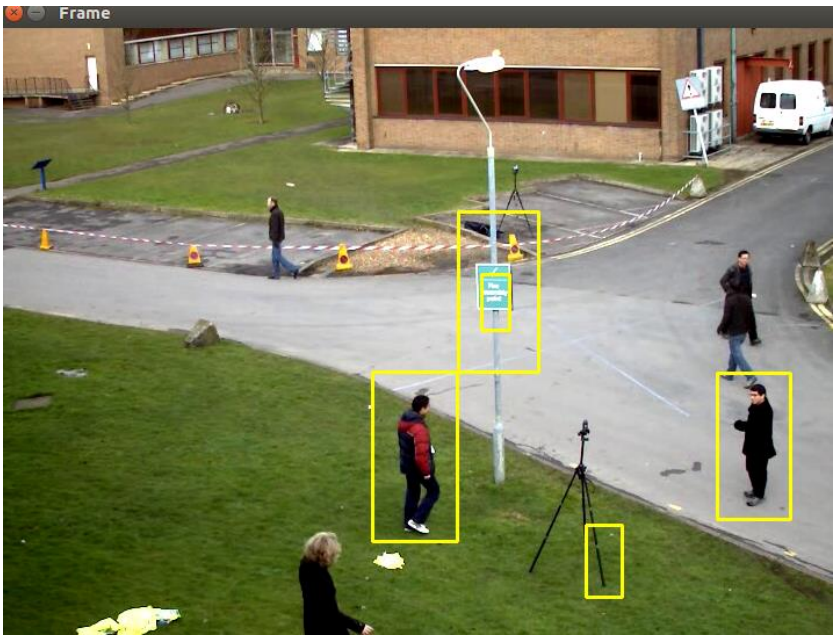


**Figura 41. Resultado 1_ PersonaHaar_ PorDefecto**

La aparición de falsos positivos es evidente, pues bien, objetos completamente diferentes: señales de tráfico, elementos de alumbrado u objetos extraños en la vía son detectados como personas. Por otra parte, la tasa de verdaderos positivos es relativamente pobre. En la captura anterior se puede ver la existencia de 6 personas y únicamente son 2 las personas detectadas como tal, esto quiere decir que la tasa de verdaderos positivos para esta captura es:

$$X = {}^{(100 * 2)}/_{6} = 33{,}33333\%$$

Calculando la tasa de verdaderos positivos para cada "*frame*" y posterior media aritmética de todos es posible conocer la tasa de acierto para el algoritmo planteado.

Aumentando las muestras utilizadas para entrenar el clasificador Haar, en este caso, 2000 muestras positivas y 1950 muestras negativas los resultados obtenidos son los siguientes:

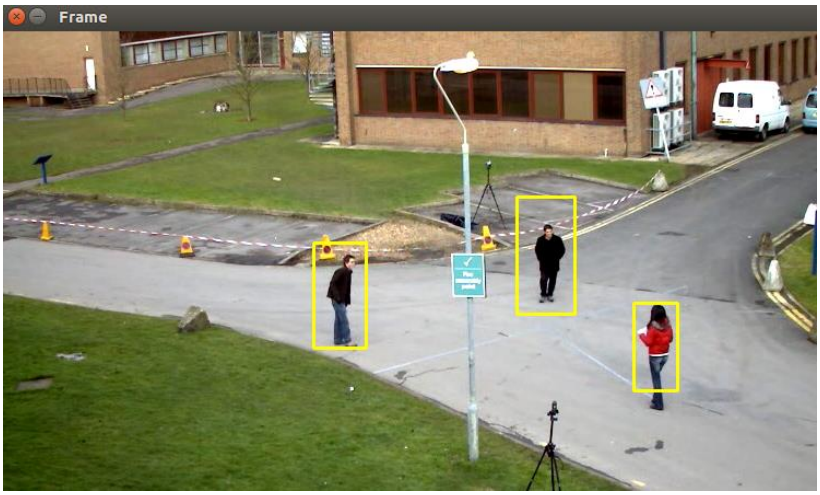


**Figura 42.Resultado 2_ PersonaHaar 2000P_1950N**

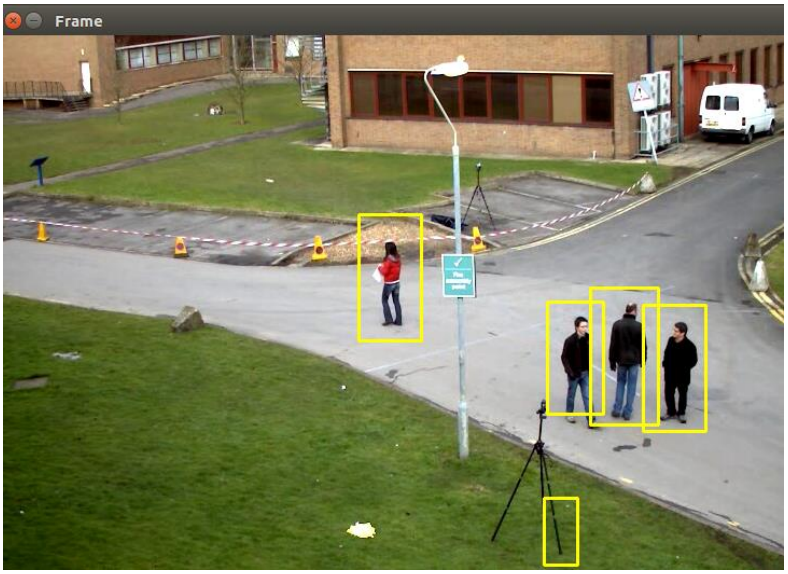


**Figura 43. Resultado 3_ PersonaHaar 2000P_1950N**

En las dos imágenes anteriores se puede notar como ha mejorado el algoritmo de manera significativa. La tasa de verdaderos positivos es muy buena, donde, se realiza una clasificación casi perfecta de los objetos presentes en la imagen. Por otro lado, la aparición de falsos positivos se ha reducido de manera notable.

A pesar del aumento de muestras, no es rara la aparición de falsos positivos de manera ocasional. Para mejorar aún más el algoritmo, esto es, reducir el número de falsos positivos y aumentar el número de verdaderos basta con mejorar el clasificador con la adición de mayor numero de muestras.

# 5 DISEÑO SISTEMA DE CONTROL DRONEKIT

En este apartado se analizan las principales funciones que proporciona la plataforma DroneKit para el diseño del algoritmo de control y manejo del dron. También se muestra la forma de crear un único "*script*" combinando el algoritmo de visión y el algoritmo de control a través de Python. DroneKit proporciona las siguientes funciones para operar con el dron:

## 5.1 Conexión con vehículo

Para conectar uno o más vehículos se va a hacer uso del método "*connect ()"*. Este método devuelve un objeto tipo "*Vehicle*" que se será el objeto a partir del cual se obtienen los parámetros y datos para controlar el movimiento del vehículo.

Un simple ejemplo de como se ha utilizado este método es:

```
from dronekit import connect
# Conectar con el vehículo (Usando puerto UDP)
Vehicle = connect ('127.0.0.1:14550', wait_ready=True)
```

El método recibe dos parámetros, el primero se corresponde con la dirección del objeto a analizar, que en función de la conexión que se pretenda establecer, se introduce una cadena u otra. La siguiente tabla muestra el tipo de conexión que se desee establecer, así como, la cadena a emplear para tal conexión.

| Tipo de Conexión | Cadena de Conexión |
|---|---|
| Ordenador Linux conectado a vehículo vía | /dev/ttyUSB0 |
| Ordenador Linux conectado a vehículo vía puerto Serie** | /dev/ttyAMA0 (baud=57600) |
| SITL conectado a vehículo vía UDP | 127.0.0.1:14550 |
| SITL conectado a vehículo vía TCP | tcp:127.0.0.1:5760 |
| Ordenador OSX conectado a vehículo vía USB | dev/cu.usbmodem1 |
| Ordenador Windows conectado a vehículo vía USB | com14 |
| Ordenador Windows conectado a vehículo vía telemetría 3DR en COM14 | com14 (baud=57600) |

**Tabla 3. Conexiones compatibles con DroneKit**

**Es el puerto al que se conecta la Raspberry Pi

El segundo parámetro se corresponde con "*wait_ready*". Este parámetro determina si la conexión se realiza inmediatamente o se espera a que se lleven a cabo ciertos parámetros y atributos. En el ejemplo mostrado, "*wait_ready=True"*, indica que se espera cierto tiempo hasta que se ponen en marcha de ciertos parámetros internos del dron. Otros parámetros que también puede recibir el método son "*baud*" o tiempo de conexión. Por último, para conectarse a diferentes vehículos basta con llamar al método tantas veces como sea necesario, cada uno, con sus respectivos parámetros de entrada.

### 5.2 Despegue

Los pasos que se deben de llevar a cabo para ejecutar la maniobra de despegue del dron son:

1. Comprobar que el vehículo se puede armar

```
While not vehicle.is_armable:
Print 'Esperar inicialización del vehículo"
time.sleep (1)
```

2. Pasar el vehículo a modelo “GUIDED”

```
vehicle.mode = VehicleMode('GUIDED)'
vehicle.armed = True
```

3. Mandar comando para armar el vehículo

```
While not vehicle.armed:
Print 'Esperar a que se arme el vehículo"
time.sleep (1)
```

4. Despegar y boquear el movimiento cuando se llegue a la altitud deseada.

```
vehicle.simple_takeoff (AltitudObjetivo)
# Restringir altura al 95% de lo establecido
If vehicle.location.global_relative_frame.alt>=AltitudObjetivo *0.95:
  print Se ha alcanzado la altura deseada"
  break
time.sleep (1)
```

El primer paso es realizar unas comprobaciones previas a través del atributo *“vehicle.is_armable*”. Este atributo encapsula información relativa al arranque, EKF y GPS propios del dron. Una salida positiva de este atributo indica que el vehículo ha arrancado, el EKF está listo y que se ha activado el bloqueo GPS.

El siguiente paso es poner el vehículo en modo “GUIDED” y armarlo. Es conveniente esperar a la confirmación de que se armado correctamente antes de mandar el comando de despegar (“*simple_takeoff ()”).* El método usado despegar es un método asíncrono por lo que puede ser interrumpido por otros comandos antes de alcanzar la altura establecida dando posibilidad de controlar y visualizar en todo momento la ubicación y altura del vehículo.

### 5.3 Mandar comandos MAVLink

Para mandar comandos MAVLink, primero se debe hacer uso del método “*message factory()*”, para codificar el mensaje y mandar el mensaje a través de la función “*send_mavlink()”.* Un ejemplo sencillo es mandar el comando: *SET_POSITION_TARGET_LOCAL_NED*. Primero se codifica el mensaje y posteriormente se envía.

```
Mensage = vehicle.message_factory.set_position_target_local_ned_encode (
```

***0,*** *# tiempo "boot" en ms*

***0, 0,*** *#Punto Sistema, Punto Componenete mavutil.mavlink.MAV_FRAME_BODY_NED,*

***0b0000111111000111,*** *# Habilita solo velocidad*

***0, 0, 0,*** *# Posiciones x, y, z*

***velocidad_x, velocidad_y, velocidad_z,*** *# Velocidad x, y, z en m/s*

***0, 0, 0,*** *# Aceleración en x, y, z*

***0, 0)*** *# yaw, yaw_rate*

Para el vehículo "COPTER" el mensaje COMMAND_LONG se puede usar para enviar una gran cantidad de comandos MAV_CMD soportados. En este caso, el envío de datos se hará:

**Mensage = vehicle.message_factory.command_long_encode (**

**0, 0,** *#Punto Sistema, Punto Componenete*

**mavutil.mavlink.MAV_CMD_CONDITION_YAW,** *#comando*

**0,** *#confirmar*

**heading,** *# Guiñada en Grados*

**0, #** *Velocidad en °/s de guiñada*

**1***, # param 3, direction -1 ccw, 1 cw*

**is_relative***, # Offset relativo, Ángulo absoluto param 4, offset 1 relativo, ángulo 0 absoluto*

**0, 0, 0)** *# 0 por defecto*

*#Mandar Comandos a Vehículo*

**vehicle.send_mavlink (Mensage)**

Existe una gran lista de comandos que pueden ser enviados al vehículo "COPTER" en modo "GUIDE". DroneKit es una API que hace abstractos gran cantidad de estos comandos. Por ejemplo:

- El método "*Vehicle.simple_takeoff ()"* proviene del comando *MAV_CMD_NAV_TAKEOFF*
- El método y atributos "*Vehicle.simple_goto()",* "*Vehicle.airspeed*" o "*Vehicle.groundspeed*" proviene del comando MAVLink *MAV_CMD_DO_CHANGE_SPEED*

## 5.4 Guía y control del vehículo. Modo "GUIDED" ("COPTER")

El modo "GUIDED" de un vuelo permite controlar el vehículo y actuar sobre el cuándo aparezcan nuevos eventos o situaciones de interés en pleno vuelo.

En este apartado se van a explicar alguno de los conceptos clave a tener en cuenta para controlar el movimiento del vehículo, así como, mandar mensajes vía MAVLink para controlar servos, orientaciones o regiones de interés.

### 5.4.1 Control Movimiento

Para controlar el movimiento del dron, se puede hacer o bien especificando una posición objetivo o bien mandando las componentes de velocidad para la dirección a la que se quiera llegar.

- Por posición objetivo:

Es útil cuando se conoce de manera predefinida la posición final. Se emplea el método "*Vehicle.simple_goto ()"*. Este método puede ser interrumpido a posteriori y tampoco indica que se haya alcanzado el objetivo.
Por otro lado, es posible definir la velocidad para alcanzar esos puntos, bien a través de atributos o la adicción de un parámetro más en la función "*simple_goto ()"*. Lo cierto es que acepta el último valor de velocidad que se asigna.
A continuación, se muestra un ejemplo de cómo establecer una posición objetivo:

***vehicle.mode = VehicleMode ('GUIDED')****#Modo de vuelo Guiado*
***Localizacion = LocationGlobalRelative (-56.245072, 132.078236, 25)***
***vehicle.simple_goto (Localizacion)***

En el ejemplo visto aparece una nueva clase que proporciona DroneKit ***LocationGlobalRelative ().*** Esta clase define un objeto de localización global en donde la latitud y longitud se hacen en correspondencia con el sistema de coordenadas WGS84 y en donde la altitud se toma en referencia con la ubicación inicial del dron (*home position*):

- ***class*** dronekit.LocationGlobalRelative (***lat***, ***lon***, ***alt=none***)

Otra clase que proporciona DroneKit es ***LocationGlobal ()***. De la misma manera que la clase anterior, se define un objeto de localización global en donde la latitud y longitud se hacen en correspondencia con el sistema de coordenadas WGS84. Ahora bien, en este caso, la altitud se toma en referencia con el nivel del mar.

- ***class*** dronekit.LocationGlobal(***lat***, ***lon***, ***alt=none***)

Otra manera de llegar a una posición predefinida es a través comandos MAVLink:
*SET_POSITION_TARGET_GLOBAL_INT*
*SET_POSITION_TARGET_LOCAL_NED*

- Acceso a posiciones por control de velocidad:

Controlar el movimiento del dron usando velocidad es más suave que usando posición cuando se tienen cambios repentinos en el entorno. La función "*send_ned_velocity ()"* genera un comando MAVLink (SET_POSITION_TARGET_LOCAL_NED) para definir los valores que adquieren las componentes de velocidad relativos a la localización "home" (MAV_FRAME_LOCAL_NED). Se debe mandar el mensaje cada segundo hasta llegar a la duración establecida.

```
Mensage = vehicle.message_factory.set_position_target_local_ned_encode (
0,       #tiempo de "boot"
0, 0, # Punto Sistema, punto componente
mavutil.mavlink.MAV_FRAME_LOCAL_NED, # Comando
0b0000111111000111,  # Solo se activa velocidad
0, 0, 0,  # Posiciones x, y, z
velocidad_x, velocidad_y, velocidad_z, # x, y, z velocidad en m/s
0, 0, 0, #Aceleración x, y, z
0, 0) # yaw, yaw_rate

#Mandar comando al vehículo a una frecuencia de 1 Hz
for x in range (0, duracion):
vehicle.send_mavlink (Mensage)
time.sleep(1)
```

Las componentes de velocidad, velocidad_x y velocidad_ y son paralelas al plano mientras velocidad_z es una componente perpendicular a ambas tomando valor positivo cuando retorna al suelo siguiendo la regla de la mano derecha.

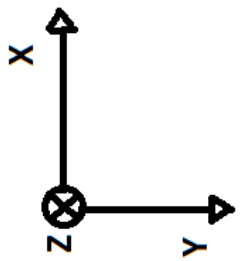


**Figura 44. Sistema de coordenadas_ Componentes Velocidad**

En definitiva, gracias a este sistema de coordenadas se puede conocer y monitorizar el movimiento de un dron. Para el caso anterior modificando los argumentos del dron de la presente forma:

- Velocidad_x > 0 indica vuelo sentido Norte
- Velocidad_x < 0 indica vuelo sentido Sur
- Velocidad_y > 0 indica vuelo sentido Este
- Velocidad_y < 0 indica vuelo sentido Oeste
- Velocidad_z > 0 indica vuelo sentido Abajo
- Velocidad_z< 0 indica vuelo sentido Arriba

### 5.4.2 Ajuste de Guiñada (YAW)

La guiñada es giro con respecto al eje perpendicular al centro del dron. Para ajustar la guiñada se manda el comando *MAV_CMD_CONDITION_YAW* codificado como mensaje *COMMAND_LONG*.

## 5.5 MISSIONES. Modo "AUTO"

### 5.5.1 Descargar misión actual

Se puede acceder a los comandos de una misión ya predefinida a través del atributo "*vehicle.commands*". Usando el método "*download*" se descargan los "*waypoints*" definidos.

```
Comandos = vehicle.commands
Comandos.download ()
Comandos.wait_ready ()
```

### 5.5.2 Limpiar misión actual

Para limpiar o eliminar la misión se emplea el método "*clear ()"*. El comando "*vehicle.commands.upload ()"* para actualizar cambios en el vehículo.

```
# Obtener Comandos del Vehículo
Comandos = vehicle.commands
# Limpiar comandos del vehículo y luego volver a cargar comandos
Comandos.clear ()
Comandos.upload ()
```

### 5.5.3 Crear y añadir nuevos comandos para misión

DroneKit permite crear y añadir nuevos comandos a una misión ya predefinida. A través del método "*add ()"* se pueden añadir nuevos comandos y son mandados al vehículo por el comando "upload ()".

```
# Obtener Comandos del vehículo
Comandos = vehicle.commands
Comandos.download()
Comandos.wait_ready ()
# Crear y añadir nuevos Comandos
ComandoA=Command (0, 0, 0,
mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT,
mavutil.mavlink.MAV_CMD_NAV_TAKEOFF, 0, 0, 0, 0, 0, 0, 0, 0, 10)
#Segundo comando
ComandoB=Command (0, 0, 0,
mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT,
mavutil.mavlink.MAV_CMD_NAV_WAYPOINT, 0, 0, 0, 0, 0, 0, 0, 10, 10, 10)
Comandos.add (ComandoA)
Comandos.add(ComandoB)
#Cargar y enviar comandos a Vehículo
Comandos.upload()
```

### 5.4.4 Modificar misiones

Hasta lo comentado, solo es posible crear y añadir nuevos comandos. Ahora bien, una operación realmente útil es modificar comandos ya existentes en una misión. Para esto, primero se crea un vector nuevo para almacenar todos los comandos de una misión, seguidamente, se hacen las pertinentes modificaciones, se eliminan los comandos iniciales y finalmente se pasa al vehículo el nuevo vector con los comandos modificados.

```
# Obtener Comandos de vehículo
Comandos = vehicle.commands
Comandos.download ()
Comandos.wait_ready ()
# Almacenar los comandos en una lista
ListaMision= []
for comando in Comandos:
        ListaMision.append (comando)
# Cambiar primer comando a MAV_CMD_NAV_TAKEOFF
ListaMision [0]. command=mavutil.mavlink.MAV_CMD_NAV_TAKEOFF
# Limpiar comandos vehículo
Comandos.clear ()
#Modificar comandos vehículo
for comando in ListaMision:
        Comandos.add (comando)
Comandos.upload ()
```

Para poder hacer el diseño completo del algoritmo de gestión de misiones autónomas, se requiere combinar el algoritmo creado para la detección y seguimiento de objetos usando la librería OpenCV y, el algoritmo de control y manejo del dron usando la librería DroneKit.

La ejecución de los dos algoritmos, tanto del algoritmo de detección y seguimiento como el algoritmo de control y manejo, se deben realizar de manera simultánea. Python permite el diseño de procesos concurrentes o multitarea. Ahora bien, para solucionar el problema de GIL ("*Global Interpreter Lock*"), un bloqueo creado por los desarrolladores de Python para garantizar que un solo hilo sea corrido al mismo tiempo por un intérprete de Python, se desarrolló el paquete *multiprocessing*, haciendo mejor uso de los subprocesos en lugar de los hilos. Este módulo permite obtener una concurrencia tanto local como remota de diferentes procesos.

Para crear un proceso a través del módulo *multiprocessing*, primero se debe hacer referencia a la clase de proceso, instanciar un objeto y pasar los argumentos. Seguidamente, se inicia el proceso a través del método *start ()* y se acaba con el método *join ()*.

```
Proceso = Process (target= ejemplo, args= ('Hola',))
Proceso.start ()
Proceso.Join ()
```

Para transferir información entre procesos y con ello establecer una comunicación entre ellos existen dos vías de comunicación:

- Colas ("*Queue*"): Se utilizan para cambiar datos entre procesos.
- Tubos("*Pipe*"): Devuelve dos objetos de comunicación, los datos mandados (método *send ()*) y los datos recibidos (método *recv ()*).

# 6 SIMULACIÓN SITL ("Software In The Loop")

Px4 es una plataforma que da la posibilidad de implementar diferentes sistemas y métodos de control para un dron. Una de las principales funcionalidades que otorga PX4 es que se pueden realizar diferentes pruebas del dron a través de simulación por medio de SITL o también a través de HITL ("*Hardware In The Loop*").

La posibilidad de simular los diferentes algoritmos de control implementados antes de realizar vuelos permite comprobar, analizar y rectificar diferentes partes del algoritmo para así evitar posibles peligros de integridad a los que puede estar expuesto el dron. Por "*Software In The Loop*" se entiende a la simulación que se lleva a cabo de un sistema sin disponer de ningún tipo de elemento Hardware de mismo.

El uso de un simulador permite que el algoritmo de control de vuelo para PX4 pueda ser testeado en un vehículo modelado, así como en un entorno también modelado. El simulador da la posibilidad de hacer interactuar el vehículo modelado, al igual que si fuese un vehículo real, con la estación en tierra, así como con diferentes sistemas embarcados. PX4 es compatible con varios simuladores, otorgando cada uno de ellos diferentes funcionalidades:

**GAZEBO**

Se trata de un simulador 3D que permite crear complejos entornos de simulación para diferentes tipos de robots autónomos, uso de visión por computador o evitar obstáculos. Es un simulador que soporta la mayoría de vehículos existentes: drones, aviones, submarinos, etc…

Para introducir en pantalla un cuadricóptero basta con ejecutar los siguientes comandos:

- *Cd ~/src/Firmware*
- *Make posix_sitl_default gazebo*

**Figura 45. Cuadricóptero_ Gazebo [27]**

A partir de aquí, es posible realizar diferentes operaciones como puede ser modificación del entorno, así como, definir nuevos parámetros al dron (ubicación, velocidades, GPS…). Cuando se usa el simulador GAZEBO se suele emplear junto con ROS, que

como ya se ha comentado en apartados anteriores, se trata de una API que permite mandar diferentes tipos de comandos para controlar el vuelo de un dron dotados con la plataforma PX4.

**JMAVSim**

Al igual que GAZEBO, se trata de un simulador usado para vehículos multi-rotor dotados de la plataforma PX4.Es un simulador que permite analizar situaciones de despegue, aterrizaje, vuelo o fallo de alguno de los sistemas embebidos en el dron. Una vez descargado e instalado el simulador, es posible ejecutar el mismo con el comando:

- *Make posix_sitl_default jmavsim*

Obteniendo la salida:


**Figura 46. Cuadricóptero_ JMAVSim [27]**

Otros simuladores más recientes y por lo tanto con menos uso son:

- AIRSIM
- XPLANE

**SIMULADOR DroneKit-SITL**

El simulador de DroneKit – SITL ("*Software In The Loop*") da la posibilidad de crear analizar y verificar aplicaciones basadas en DroneKit sin la necesidad de disponer de un vehículo real. Se trata de un simulador sencillo, simple y veloz, con posibilidad de ser lanzado en Windows, Mac o Linux.

Uno de los inconvenientes que presenta este simulador es que dispone de un número reducido de vehículos compatibles, pues bien, se trata de herramienta relativamente reciente.

Para instalar y hacer uso de este simulador basta con ejecutar el siguiente comando:

- *Pip install dronekit-sitl*

Se puede decir que el simulador se puede instalar o bien en mismo ordenador que DroneKit o bien en un ordenador alternativo conectado a la misma red. A continuación, se muestra los comandos básicos.

- *Dronekit-sitl copter --home=--23.675789, 154.765986, 543,234*

En este comando se ha definido la ejecutar la última versión de un vehículo "*COPTER*", así como, también se ha definido la ubicación inicial del vehículo. A partir de aquí, el simulador espera a alguna conexión TCP a través del puerto 127.0.0.1:5760. Es posible conectarse al simulador a través del código Python usando DroneKit:

- *Vehicle = connect ('tcp: 127.0.0.1:5760', wait_ready=True)*

Si el puerto esta cogido, el simulador espera a nuevas conexiones para los puertos 5763, 5766, 5769,5772…

Para depurar los resultados de los "scripts" o algoritmos creado, es de gran interés visualizar los resultados a través de una estación en tierra ya sea QGroundControl, MissionPlanner, o cualquier otra. Para llevar a cabo esto es necesario descargar Mavproxy, abrir una terminal nueva generando una instancia de MAVproxy para reenviar mensajes desde TCP 127.0.0.1:5760 a otros puertos UDP. Una vez creados estos puertos UDP es posible conectarse a través de uno de ellos con la estación en tierra, así como, conectarse a través desde el algoritmo de DroneKit.

**Entorno de simulación SITL para PX4**

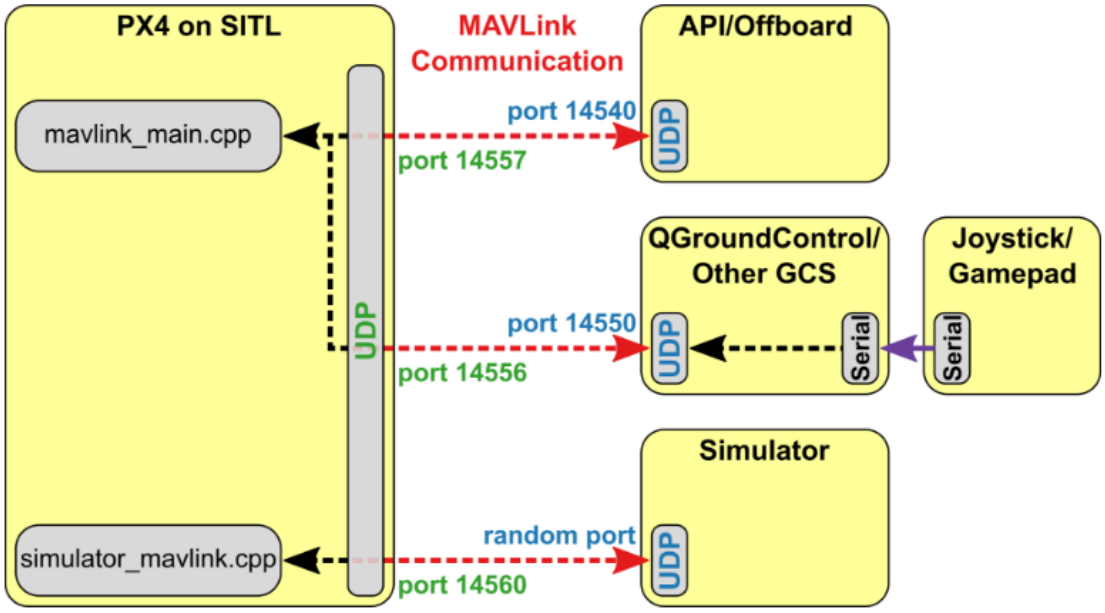


**Figura 47. Arquitectura Simulación SITL [27]**

El diagrama anterior se corresponde a la arquitectura que modela un entorno tipo de simulación SITL. En la figura, se puede apreciar que todos los componentes que forman el sistema se encuentran conectados vía UDP, solo la conexión entre la estación de tierra y el Joystick se realiza a través de una conexión serie.

Para conectar diferentes estaciones de tierra, así como, APIs externas como pueden ser ROS o DroneKit a la plataforma PX4 se usa comunicación MAVLink.

Otro hecho a destacar es que la plataforma PX4 usa un módulo específico para analizar el puerto UDP 14560, que es el puerto al que va conectado alguno de los diferentes

simuladores analizados anteriormente, QGroundControl para este proyecto. Una vez se halla conectado el simulador a este puerto, para intercambiar información con el entorno se hace uso de MAVLink API, esta API permite a través de mensajes MAVLink que la información obtenida de los sensores en el entorno simulado pase a la plataforma PX4 y que a su vez esta devuelva los valores de los diferentes actuadores a partir del algoritmo de vuelo al vehículo y entorno simulado.

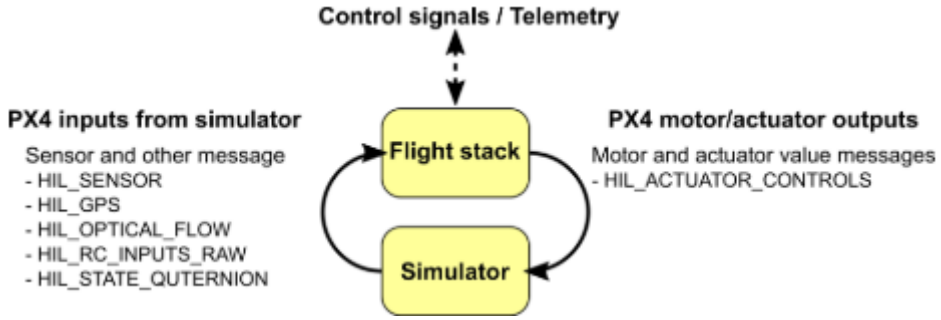


**Figura 48. Intercambio Información PX4 y Simulador [27]**

En relación a todo lo anterior, se puede decir que el puerto UDP 14540 se utilizar para comunicar la plataforma PX4 a las APIs externas, el puerto UDP 14550 se usa para establecer comunicación con las estaciones en tierra y, por último, el puerto 14560 establece comunicación con el simulador escogido.

# 7 RESULTADOS OBTENIDOS

En este punto de la memoria se hace especial énfasis en el algoritmo global diseñado para la gestión de vuelos autónomos. A continuación, se muestra la máquina de estados diseñada para llevar a cabo el algoritmo diseñado.

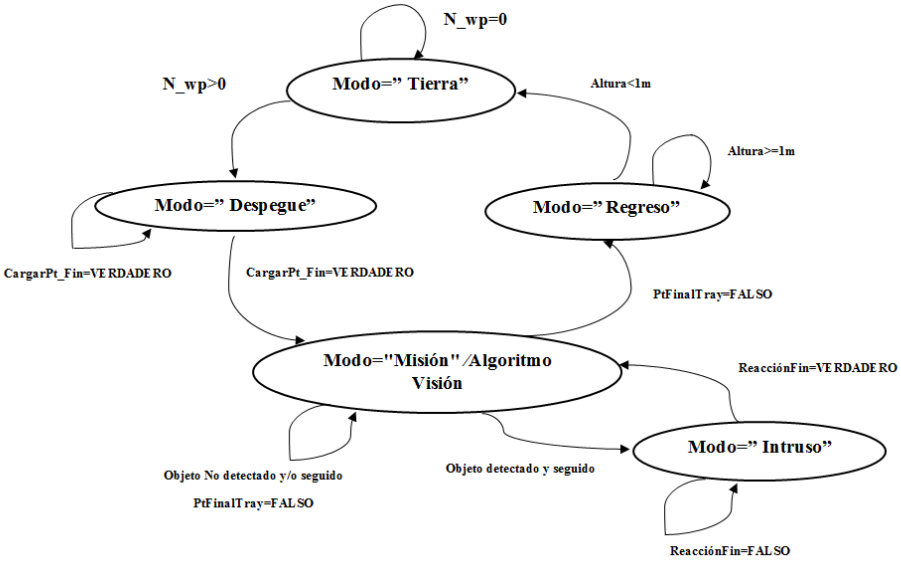


**Figura 49. Máquina de estados _ Algoritmo Gestión Vuelos Autónomos**

Para entender con más detalle el algoritmo final planteado se propone el siguiente diagrama de flujo.

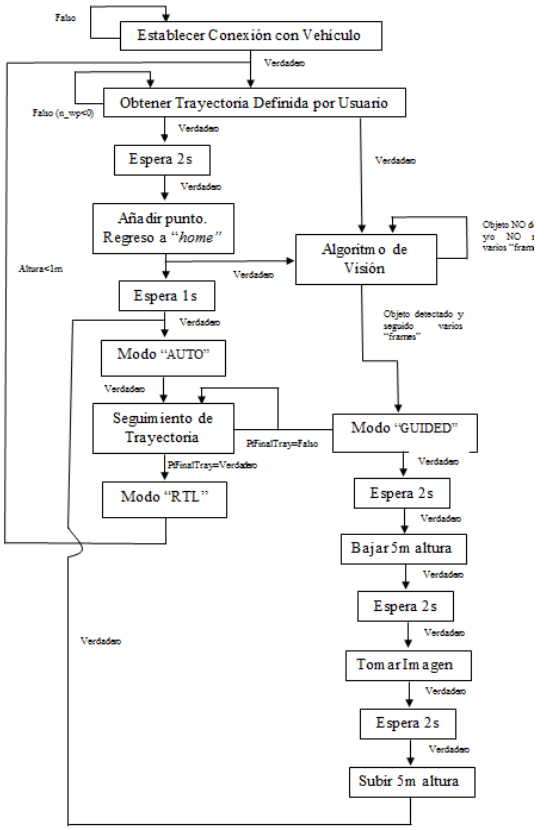


**Figura 50. Diagrama de flujo_ Algoritmo Gestión Vuelos Autónomos**

Para abordar la simulación, primero se pone en marcha el simulador de DroneKit, estableciendo la última versión de un cuadricóptero como vehículo simulado, y fijando una posición inicial para el mismo. En este caso, por motivos de simulación, se ha decidido establecer una ubicación inicial (*home*) aleatoria.

- ***Dronekit-sitl copter home=67.766294, -6.645524, 0,0***


**Figura 51. Puesta en Marcha DroneKit-SITL**

Seguidamente, por medio de MAVproxy, abrir una terminal nueva generando una instancia de MAVproxy para reenviar mensajes desde TCP 127.0.0.1:5760 a otros puertos UDP, a donde conectar otros dispositivos o plataformas como pueden ser estaciones en tierra o dispositivos embarcados.

- ***Mavproxy.py --master=tcp:127.0.0.1:5760 --out=udp:127.0.0.1:14550 --out=udpout:127.0.0.1:14549***

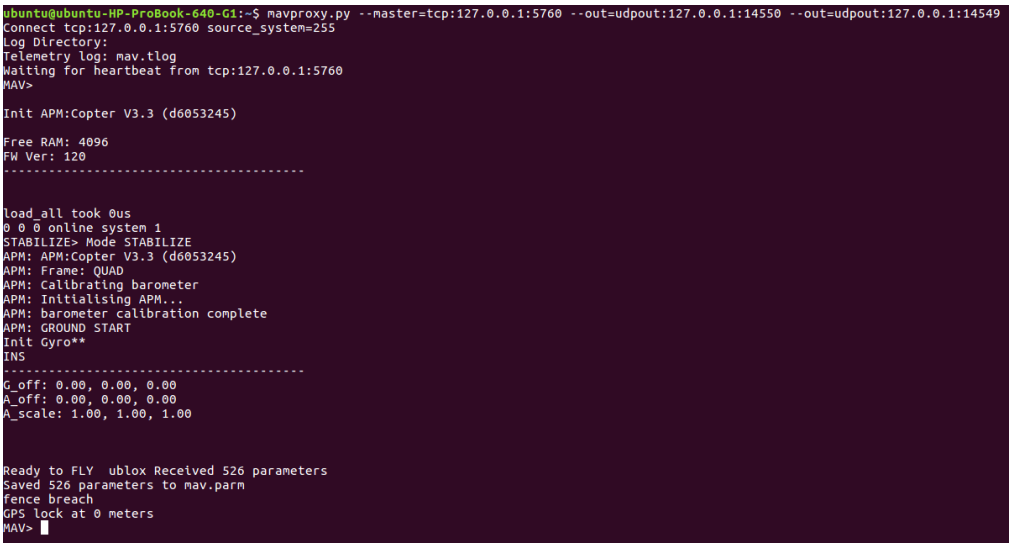


**Figura 52. Puesta en Marcha Comunicación MAVLink_ MAVproxy**

Una vez lanzado el comando anterior, de manera automática, aparece la visualización del vehículo a simular a través de QGroundControl.

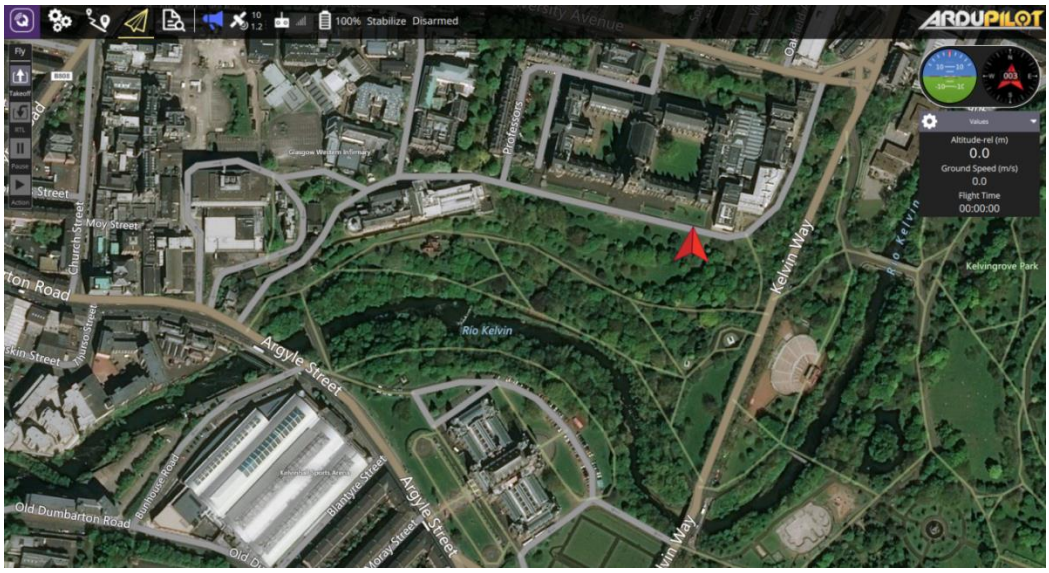


**Figura 53. Interfaz QGroundControl_ Vehículo Simulado**

Por último, se lanza en algoritmo de gestión de misiones autónomas a través del comando:

- ***Sudo Python FinalAuto.py***

```
ubuntu@ubuntu-HP-ProBook-640-G1:~$ cd Escritorio
ubuntu@ubuntu-HP-ProBook-640-G1:~/Escritorio$ cd Video_Car_Pedestrian/
ubuntu@ubuntu-HP-ProBook-640-G1:~/Escritorio/Video_Car_Pedestrian$ ls
AUTO1.py         Face.py                    FinalPedestrianHaarCascade.xml
car.py           FinalAUTO.py               outVideo.avi
carretera1.avi   FinalBodyHaarCasacade.xml  pedestrian.py
carretera2.avi   FinalCarHaarCascade.xml    pedestrians.avi
cars3.xml        FinalFaceHaarCascade.xml   Proceso.py
ubuntu@ubuntu-HP-ProBook-640-G1:~/Escritorio/Video_Car_Pedestrian$ sudo python FinalAUTO.py
[sudo] password for ubuntu:
Connecting...
>>> APM:Copter V3.3 (d6053245)
>>> Frame: QUAD
Downloading mission
Downloading mission
Downloading mission
Downloading mission
```

**Figura 54. Lanzamiento Algoritmo Gestión Vuelos Autónomos**

Para probar que el algoritmo diseñado cumple con los objetivos iniciales se va a realizar la simulación de una misión cualquiera definida por el usuario. En este caso la trayectoria simulada dispone de las siguientes características:

| Trayectoria | Altura(m) | Trayectoria | Altura(m) |
|---|---|---|---|
| home | 0 | Punto 6 | 50 |
| Punto 1 | 50 | Punto 7 | 50 |
| Punto 2 | 50 | Punto 8 | 50 |
| Punto 3 | 50 | Punto 9 | 50 |
| Punto 4 | 50 | Punto 10 | 50 |
| Punto 5 | 50 | Punto Inicial | 50 |

**Tabla 4. Características Trayectoria Simulada**

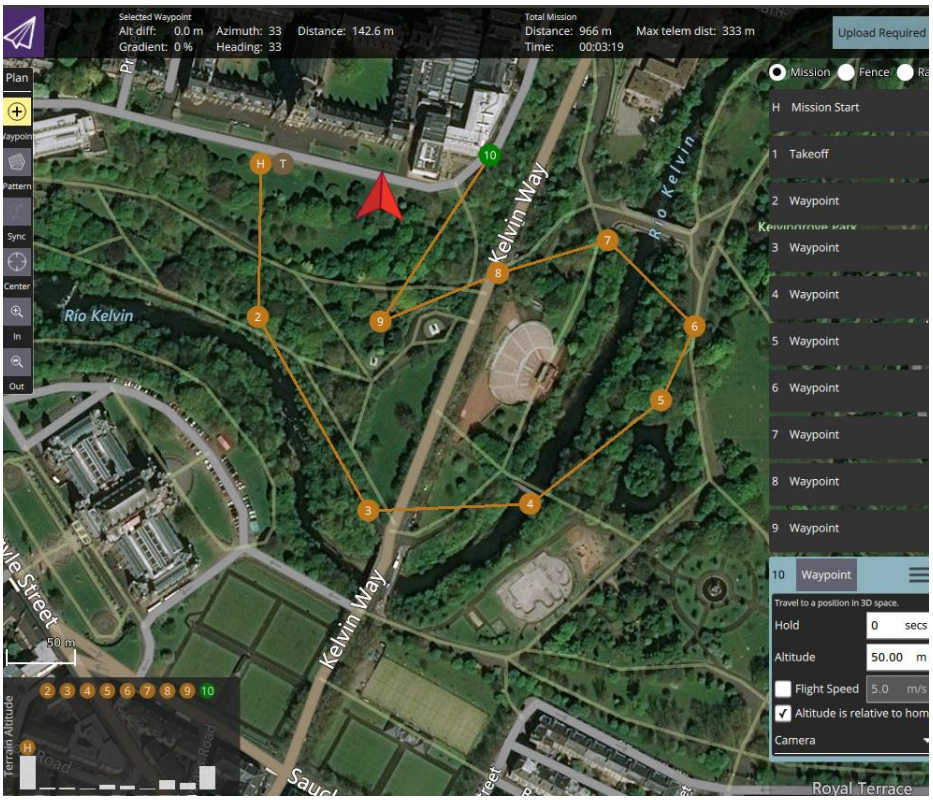


**Figura 55. Trazado Trayectoria Simulada**

En la captura anterior se puede ver como la trayectoria definida es completamente aleatoria, es decir, el algoritmo responde ante cualquier trazado planteado. Por otro lado, en cuanto a la altura de los puntos se ha mantenido la altura por defecto que otorga QGroundControl a los puntos creados.

La simulación se puede hacer con la detección de cualquiera de los objetos entrenados, esto es, vehículos, personas o caras. Ahora bien, para hacer la simulación más real, en este caso, se ha decido realizar la detección y seguimiento en tiempo real de caras para la gestión de las misiones. Pues bien, es lo más cerca posible para ajustar la simulación SITL al entorno real. En las siguientes capturas se muestra el flujo de operación del sistema diseñado.

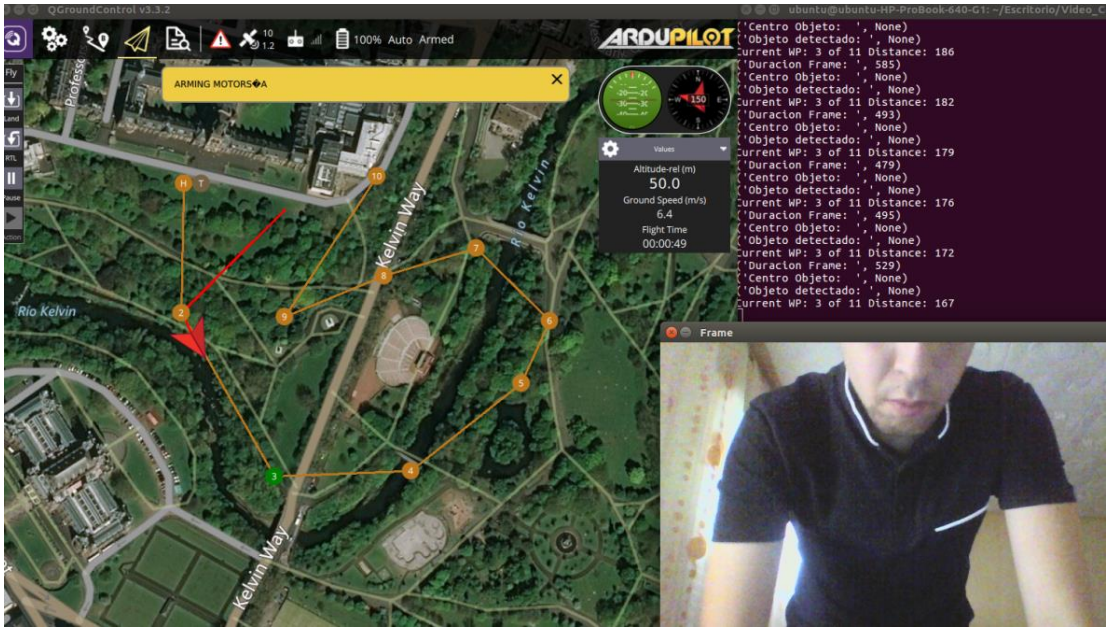


**Figura 56. Resultado 1_ Algoritmo Gestión Vuelos Autónomos**

En esta primera captura se puede observar como el vehículo modelado empieza a trazar la trayectoria definida al mismo tiempo que va realizando un estudio del entorno en busca de objetos de interés, caras en este caso. Para detectar la cara, como es de esperar, se tiene que ver completamente. Es por ello que la cara vista parcialmente no se detecta como cara.

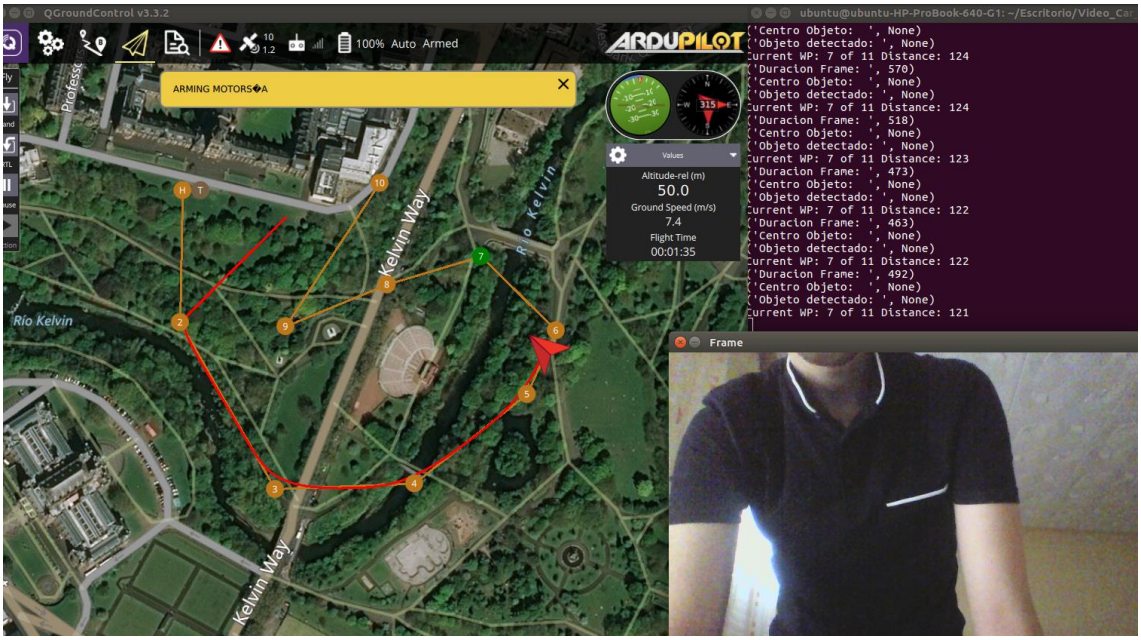


**Figura 57. Resultado 2_ Algoritmo Gestión Vuelos Autónomos**

El análisis del entorno se hace de manera simultánea y en tiempo real junto con el trazado de la misión. Una vez se detecta y se sigue la presencia de un objeto intruso durante cierto número de "*frames*" (10), es este caso modelado a través de una cara, se

lleva a cabo la reacción por parte del dron y se entra en modo ‘Intruso’. La reacción como se puede ver es:

- Parar la misión en punto donde se conoce con exactitud la presencia del objeto.
- Estabilizarse en esa posición durante 2 segundos.
- Bajar 5 metros en altura para visualizar mejor el objeto detectado.
- Estabilizarse en esa posición durante 2 segundos.
- Subir 5 metros en altura para volver al punto de partida
- Estabilizarse en esa posición durante 2 segundos.
- Retomar la misión

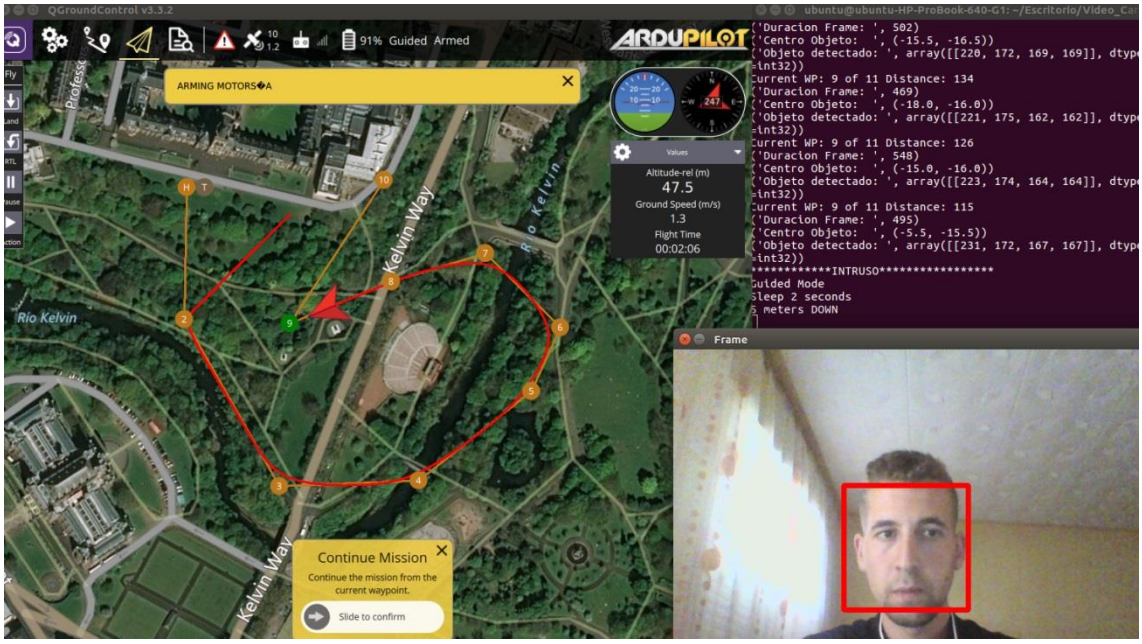


**Figura 58. Resultado 3_ Algoritmo Gestión Vuelos Autónomos**

Efectivamente se puede observar que la altura en función del tiempo seguida por el dron, para una altura inicial de 50 metros, es la siguiente:

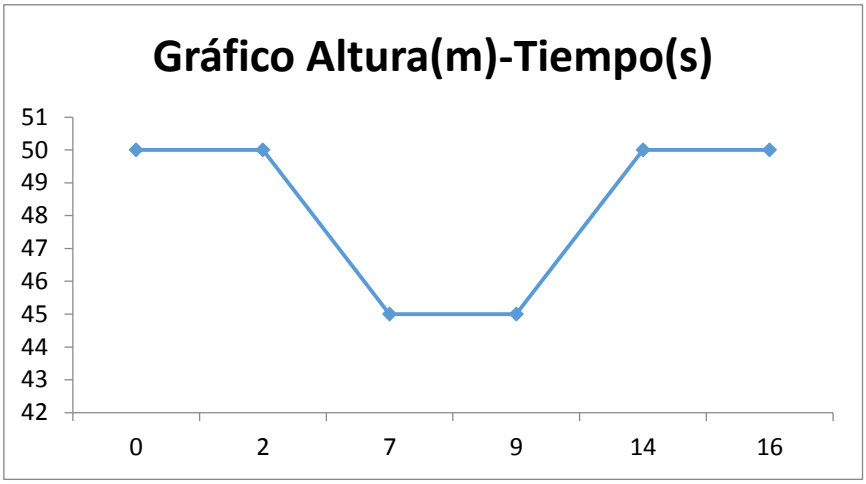


**Figura 59. Gráfico Altura-Tiempo: Reacción del Dron**

Una vez el objeto desaparece, no se detecta su presencia o simplemente es un falso positivo (no se puede hacer un tracking o seguimiento), el dron retoma la misión en el mismo punto que se había dejado en busca de nuevos intrusos, en este caso caras.

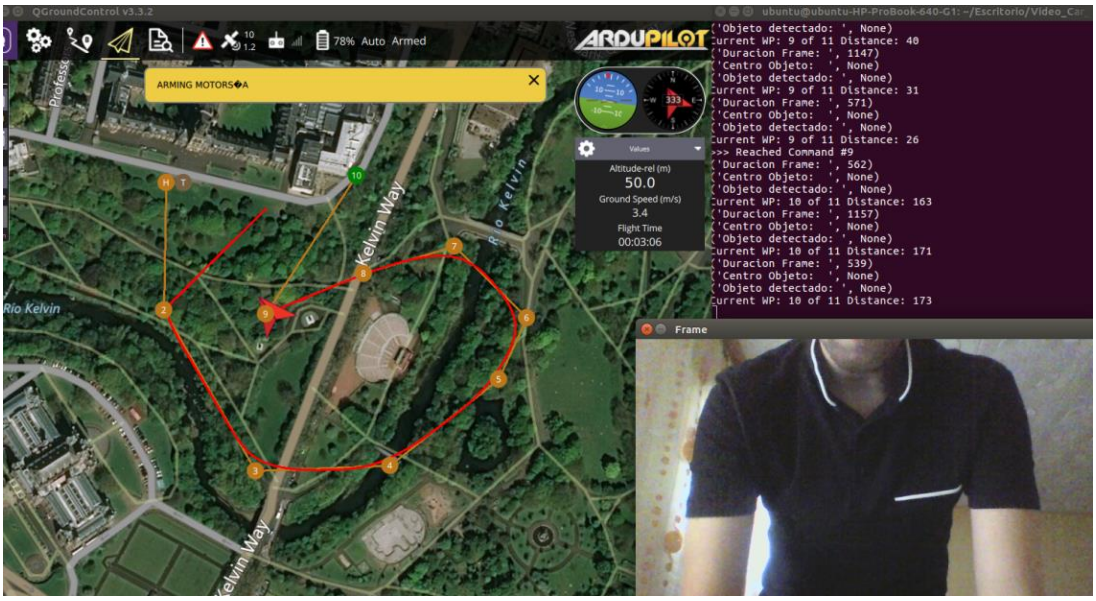


**Figura 60. Resultado 4_ Algoritmo Gestión Vuelos Autónomos**

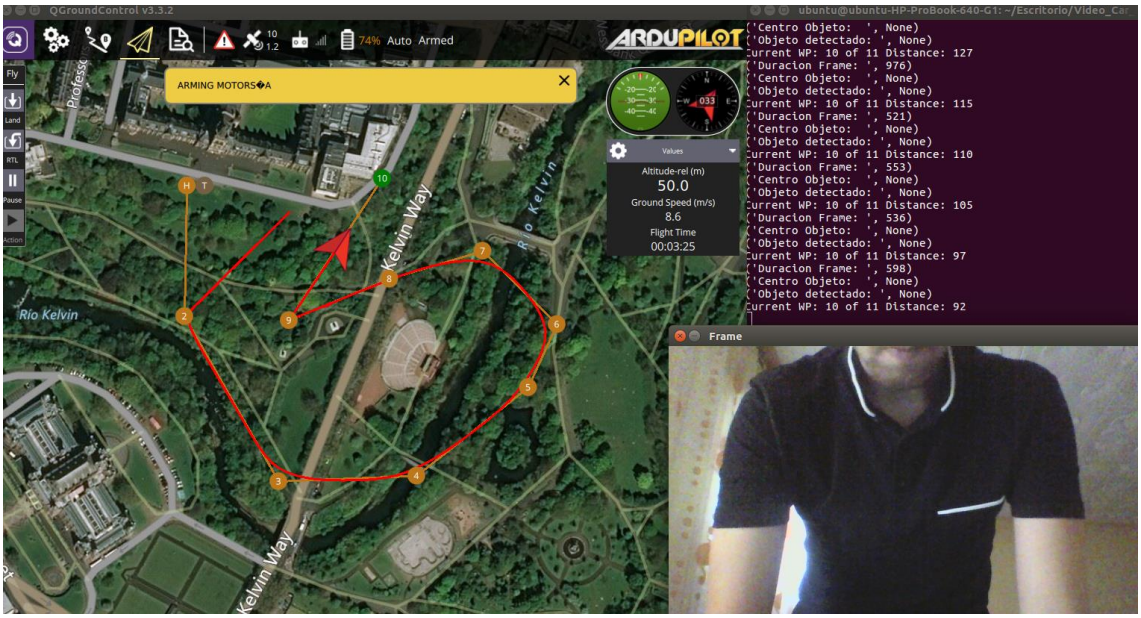


**Figura 61. Resultado 5_ Algoritmo Gestión Vuelos Autónomos**

Una vez se alcanza el último punto definido por el usuario se vuelve al punto de partida inicial tal como puede verse en la siguiente figura.

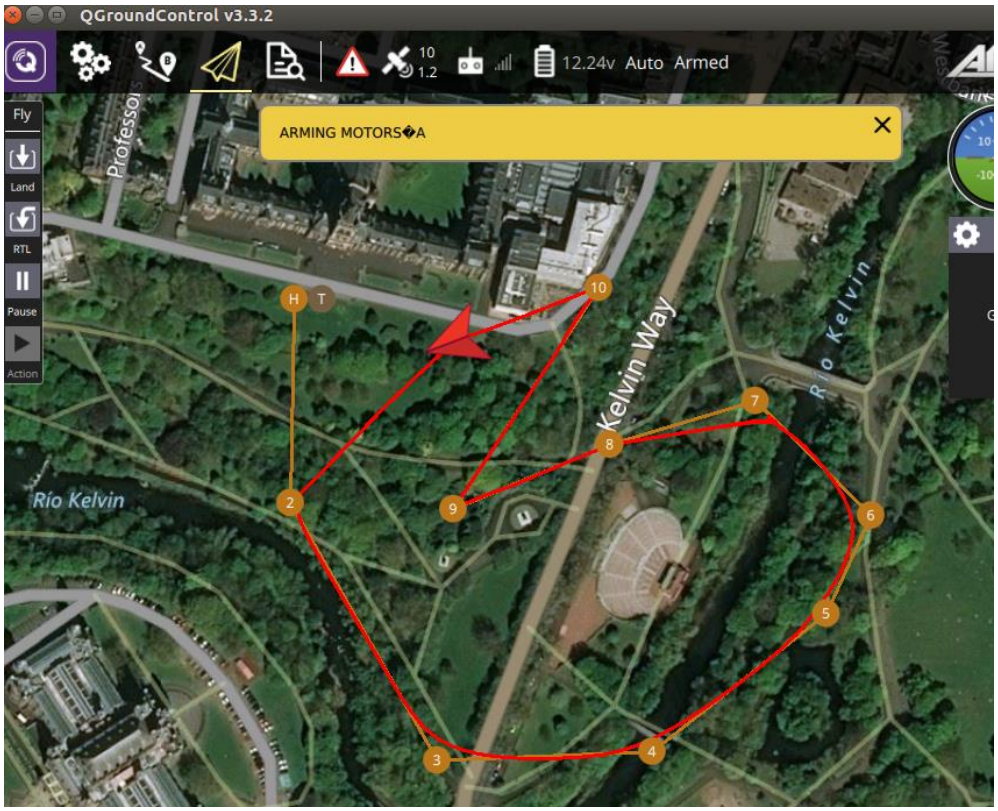


**Figura 62. Resultado 6_ Algoritmo Gestión Vuelos Autónomos**

Una vez comentado esto, se ha decidido lanzar el algoritmo de gestión de misiones autónomas desde la Raspberry Pi y ver que efectivamente se obtiene el mismo resultado que cuando se lanza desde el ordenador convencional. Para llevar a cabo lo planteado se ha conectado a la misma red tanto a la Raspberry Pi como al simulador como a la estación en tierra.

Primero nos conectamos a la Raspberry de forma remota, tal como se ha comentado en apartados anteriores.

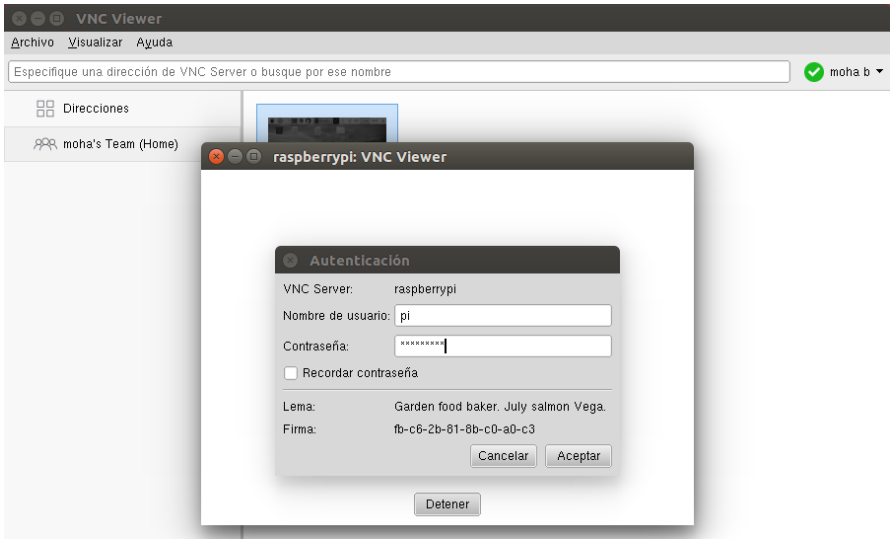


**Figura 63. Acceso Remoto Raspberry Pi**

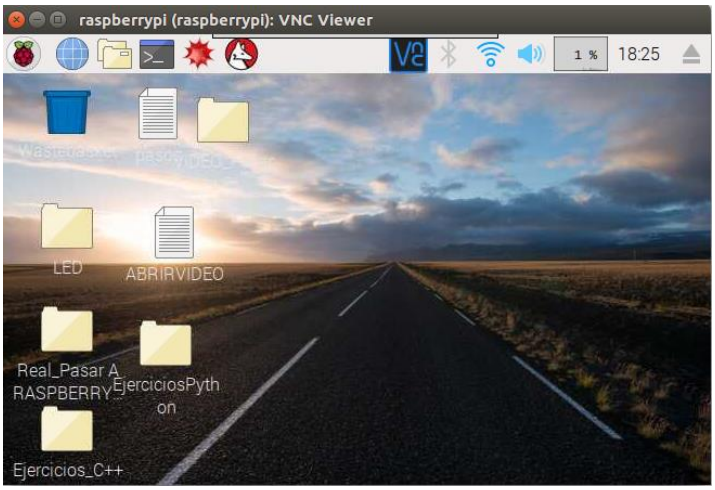


**Figura 64. Escritorio Raspberry Pi**

Una vez se ha accedido a la Raspberry Pi, el siguiente paso es, al igual que antes, lanzar el simulador DroneKit-SITL.

- ***Dronekit-sitl copter home=64.760956, -6.135543, 0,0***

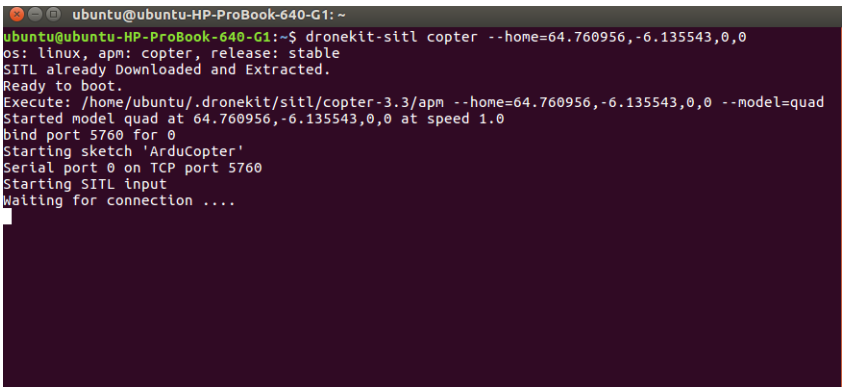


**Figura 65. DroneKit-SITL**

Siguiendo el mismo planteamiento que antes, se abre una terminal nueva ligada con Mavproxy. En este caso se ha añadido un puerto UDP adicional que usa la Raspberry Pi.

***Mavproxy.py --master=tcp:127.0.0.1:5760 --out=udp:192.168.43.17:14550 --out=udp:127.0.0.1:14550 --out=udpout:127.0.0.1:14549***

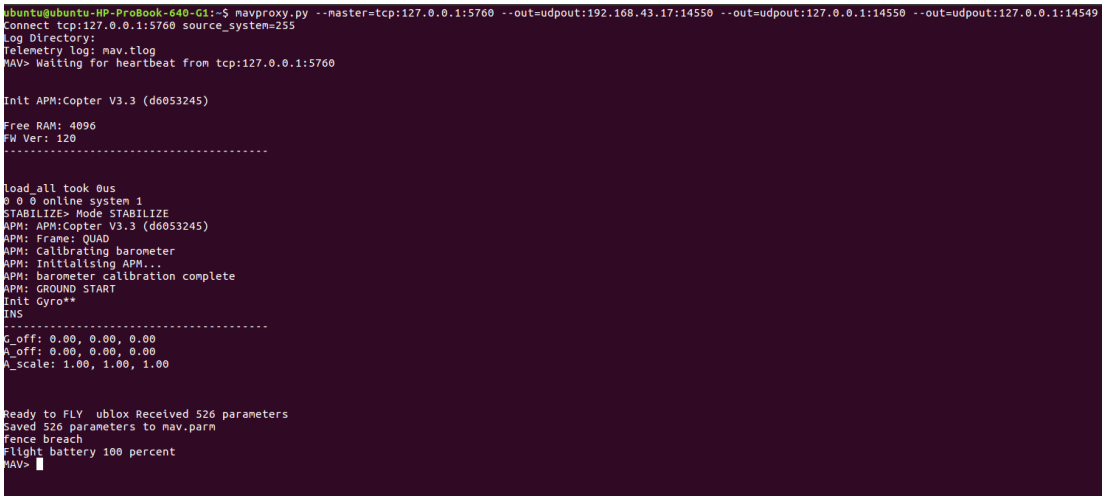


**Figura 66. MAVLink_ MAVproxy**

De manera similar, QGroundControl iniciará de manera automática:

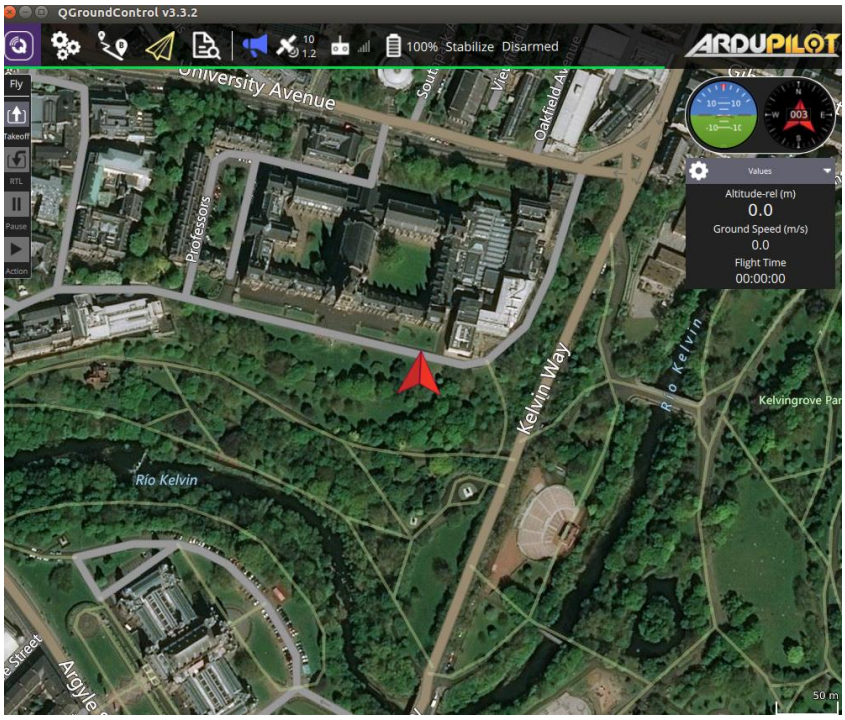


**Figura 67. QGroundControl**

Finalmente se lanza el algoritmo diseñado en la Raspberry Pi y se puede comprobar que el algoritmo funciona de manera idéntica a la simulación anterior.

- ***Sudo Python FinalAuto.py***

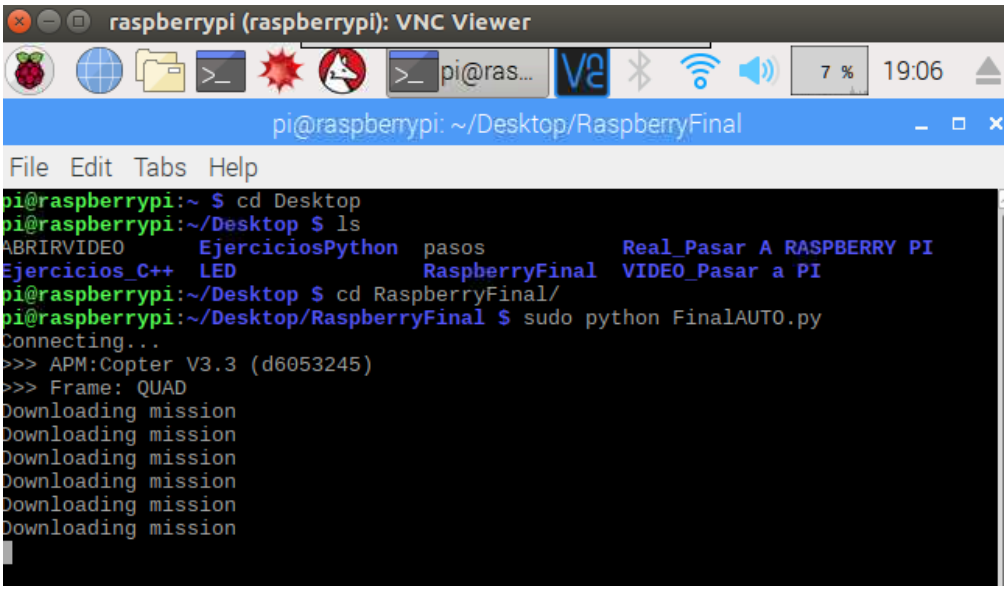


**Figura 68. Lanzamiento Algoritmo Gestión de misiones autónomas desde Raspberry Pi**

Por último, se puede comentar que una vez se detecten objetos de interés, en este caso caras estas se almacenaran en formato .png en una carpeta complementaria al algoritmo de gestión de misiones autónomas.

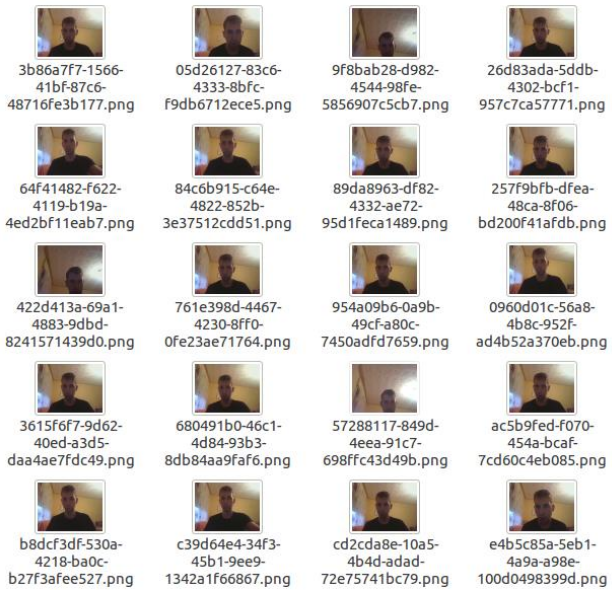


**Figura 69. Captura de objetos sospechosos**

# 8 SIMULACIÓN HITL (“Hardware in the Loop”)

En este octavo apartado de la memoria se muestran los pasos que se han seguido para comprobar que es posible implementar el algoritmo de manipulación de las misiones en la Raspberry Pi y que a su vez es posible interactuar con la controladora de vuelo Pixhawk.

## 8.1 Comunicación Raspberry Pi 3 modelo B con Pixhawk

Tal y como se ha ido comentado a lo largo de este proyecto, la Raspberry Pi dota al sistema embarcado en el dron de una serie de funcionalidad que con el uso de únicamente la controladora de vuelo sería imposible de llevar a cabo.

En este proyecto es de vital importancia pues contribuye a procesar una ingente cantidad de imágenes mejorando el flujo en memoria de manera considerable.

Por todo lo dicho se hace evidente establecer una comunicación estable y robusta entre Pixhawk y Raspberry Pi. La comunicación entre ambos dispositivos se lleva a cabo a través de una conexión serie usando MAVLink como protocolo de comunicaciones.

1. Conexión Física

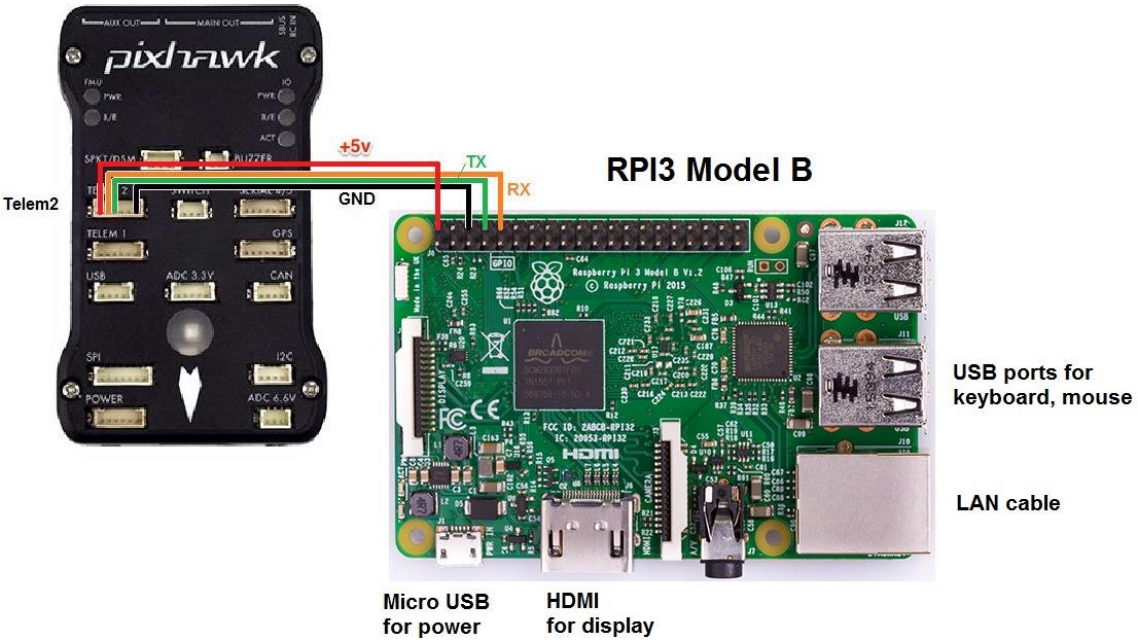


**Figura 70. Conexión Física Pixhawk_ Raspberry Pi 3 modelo B [4]**

Como se puede ver en la imagen, el puerto de telemetría secundario es el bus de pines de interés. Este bus de pines se compone de seis pines distribuidos de esta manera:

**(1) 5V; (2) TX; (3) RX; (4)CTS; (5) RTS;**

Para establecer una conexión en serie con la Raspberry Pi, se requiere hacer uso de los pines **TX**, **RX** y los pines **GND** y **5V** de alimentación. Las conexiones se realizan de tal manera que se muestra en la figura. Tal como se comentó en el apartado de acceso a los pines de la Raspberry Pi, los pines para una conexión serie corresponden a los pines **GPIO14 (TX), GPIO15 (RX).**

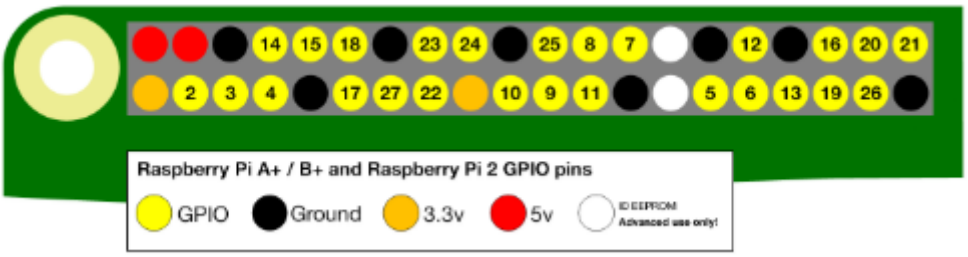


**Figura 71. Pines Entrada-Salida (GPIO) Raspberry Pi 3 Modelo B**

2. Configuración Raspberry Pi

En primera instancia, para evitar la transmisión de datos innecesarios por el puerto serie, se debe deshabilitar la salida de consola del sistema operativo que justamente pasa por este puerto por defecto. Para ello:

- *Sudo raspi-config*

En la opción 8("**Advanced Options"**) se selecciona la opción 7("**Serial"**) y se configura como **No**.

Configuración puerto serie Raspberry Pi

- *Sudo nano /boot/config.txt*

Al final del archivo se añade *enable_uart*=1
Para utilizar el puerto dev/ttyAMA0 para la conexión serie también es necesario incluir al final del archivo *dtoverlay=pi3-minuart-bt*
Para comprobar que se ha realizado correctamente la configuración:

- *ls -1 /dev*

Configuración puerto serie Pixhawk

Se conecta la Pixhawk con una estación en tierra y se hablita la comunicación MAVLink a través del puerto serie, para ello se ajusta el parámetro:

- *SERIAL2_PROTOCOL = 1*

A continuación, para establecer la comunicación vía MAVLink se instalan estos paquetes en la Raspberry Pi:

- *Sudo apt-get update*
- *Sudo apt-get install screen Python-wxgtk2.8 python-matplotlib python-opencv python-pip python-numpy python-dev libxm12-dev libxslt-dev python-lxml*
- *Sudo pip install future*
- *Sudo pip install pymavlink*
- *Sudo pip install mavproxy*

Para testear que ambos dispositivos se encuentran comunicados se puede hacer uso del comando:

- *Sudo –s*
- *Mavproxy.py –master=/dev/ttyAMA0 –baudrate 57600 –aircraft mydrone*

Mediante este comando se establece una conexión por el puerto ttyAMA0 a una tasa de 57600 baudios usando Mavproxy.py
Para finalizar se tiene que verificar que no existe ningún tipo de error de comunicación entre la emisora y el dron se pueden realizar una serie de pruebas.

- *Param show ARMING_CHECK*
- *Param set ARMING_CHEK 0*
- *Arm throttle*

Para configurar la ejecución automática de Mavproxy siempre que se arranque la Raspberry Pi en el archivo /etc/rc.local y se cambia el texto de la siguiente manera.

- *Sudo nano /etc/rc.local*
- *(*
  *Date*
  *Echo $PATH*
  *PATH=$PATH:/bin:/sbin:/usr/bin:/usr/local/bin*
  *Export PATH*
  *Cd/home/pi*
  *Screen –d-m-s /bin/bash mavproxy.py –master=/dev/ttyAMA0 –baudrate 57600 –aircraft mydrone*
  *) > /temp/rc.log 2>&1*
  *Exit 0*

### 8.2 Lanzamiento del Algoritmo al arrancar Raspberry Pi

Lo primero que se hace es configurar la Raspberry Pi de tal manera que arranque DroneKit noventa segundos después de que se haya iniciado la Raspberry Pi y que se lleve a cabo el algoritmo desarrollado. Para ello basta ejecutar el siguiente código siempre que se encienda la Raspberry Pi:

1. Se crea algoritmo arranque.sh

- *Sleep 90*
  *Cd/ home/Pi/Python*
  */user/bin/Python FinalAUTO.py --connect /dev/serial0*

2. Ejecutar código siempre que se arranque Raspberry Pi

- *Crontab –e*
  *@reboot /root/arranque.sh >> /root/dronekit_scrip.log 2>&1*

# 9 ESTUDIO SOCIO-ECONÓMICO

## 9.1 Presupuesto

En este capítulo de la memoria se va a realizar, de forma aproximada, el cálculo del coste total que lleva la elaboración de este proyecto. Se estima que el tiempo que se ha requerido para la elaboración del proyecto ha sido de 7 meses a una jornada de 5 horas de lunes a viernes.

$$TotalHorasProyecto = 5\ {}^{horas}/_{dia} * 5\ {}^{dias}/_{semana} * 5\ {}^{semanas}/_{mes} * 7\ meses = 700\ horas$$

Para satisfacer económicamente el trabajo realizado por el ingeniero, se va a suponer que el salario bruto medio anual para un ingeniero software recién graduado es de 20.000 euros. Considerando condiciones nominales del trabajador de estas características (recién graduado, soltero, sin ascendientes y descendientes a cargo).

| | **Año 2018** |
|---|---|
| Sueldo Bruto Anual | 20.000,0€ |
| Retenciones IRPF | 2.338,0 € |
| Cuotas Seguridad Social | 1.270,0 € |
| Tipo de Retención sobre nómina | 11.69 % |
| Sueldo neto Anual | **16.392,0 €** |

**Tabla 5. Desglose Sueldo neto Anual**

Teniendo en cuenta que para el año 2018 el número de días laborables es de 251 días entonces:

$$HorasTrabajo2018 = 251\ días * 8\ {}^{horas}/_{dia} = 2008\ {}^{horas}/_{Año}$$

Por todo ello, los honorarios que corresponden al ingeniero son:

$$SueldoNeto_{700horas} = \frac{700}{2008} * SueldoNeto_{2008horas} = \frac{700}{2008} * 16.392{,}0 = 5.714{,}35\ €$$

Por otro lado el desglose de adquisición de material es:

| **Material** | **Precio** |
|---|---|
| HP Probook 640 G1 | 375,0 € |
| Raspberry Pi modelo B | 59,90 € |
| Dron 4 rotores | 680,0€ |
| Material complementario | 29,80 € |
| | **1.144,7 €** |

**Tabla 6. Desglose Coste Total Material**

En definitiva el coste total del proyecto será:

$$Cos\ Total = HonorariosIngeniero + CosteMaterial = 5.714{,}35 + 1.144{,}7 = \mathbf{6.859{,}05}€$$

## 9.2 Impacto Social

La seguridad de los usuarios es un tema con gran impacto. Con la entrada de la transformación digital el gran objetivo de las grandes empresas y desarrolladores es mejorar el nivel de seguridad en todos los ámbitos de la vida, teniendo como medio para ello, el desarrollo continuo de la tecnología. En este sentido, el enfoque que se da a casi cualquier proyecto tecnológico está en conseguir aumentar el nivel de satisfacción y seguridad del cliente.

Áreas como el desarrollo de nuevos sistemas de transportes, como es la conducción completamente autónoma, ya sea, en vehículos de tráfico, ferrocarriles o vehículos aéreos están orientados principalmente en eliminar el fallo humano a través de tecnología con grandes prestaciones. Así también, la búsqueda de nuevas soluciones para mejorar y aumentar la cyberseguridad está a la orden del día, pues bien se sabe que, la expansión de internet continua creciendo de manera exponencial. .

A través de este proyecto se busca aportar una estrategia adicional para aumentar la seguridad de recintos privados ante posibles eventos de peligro y mejorar la vida privada de las personas. El empleo del sistema propuesto en combinación con los existentes sistemas de vigilancia permite alcanzar altos niveles de seguridad para este tipo de recintos. El sistema diseñado está encaminado hacia recintos con pocas aglomeraciones de personas, por lo que, ante cualquier entrada sospechosa el sistema dispondrá de las capacidades necesarias para interactuar.

Por otro lado, por medio del presente proyecto se busca introducir en la sociedad a los drones como un componente más del día a día e intentar eliminar el rechazo que alguna parte de la población pueda tener acerca de estos vehículos no tripulados.

# 10 CONCLUSIÓN Y TRABAJO FUTURO

Por medio del presente proyecto se ha logrado demostrar la posibilidad que existe de poder interactuar con un vuelo autónomo de un dron basado en el seguimiento de diferentes *waypoints* GPS. Se ha conseguido, a través, de visión artificial hacer un análisis continuo del entorno recorrido y en función de ello dotar al dron de las capacidades necesarias para tomar decisiones por sí mismo.

La solución planteada es una primera aproximación para obtener una solución con mayor modularidad y que permita ser utilizada en diferentes entornos, cambios de iluminación y diferentes objetos a detectar.

Como se ha comentado, para conseguir alcanzar resultados más punteros y exactos se hace necesario el empleo de gran cantidad de información y con ello sistemas de cómputo con características técnicas superiores a las que solo una Raspberry Pi puede hacer frente. En este sentido, el empleo de técnicas de detección y seguimiento más sofisticadas como pueden ser "*Deep Learning*" permitiría obtener resultados exponencialmente mejores a los obtenidos a través de clasificadores Haar.

En relación a los tres métodos de detección planteados para la elaboración de este proyecto: Descriptor HOG + SVM; Características Haar+ Clasificador en Cascada; Características LBP + Clasificador en Cascada; es el método basado en características Haar y empleo de un clasificador en cascada es el que me mejor resultados proporciona, y que por las limitadas características del dispositivo de computo (Raspberry Pi), se ha implementado como sistema de detección final por su poca carga en memoria y por eficaz puesta en marcha.

Para este sistema de clasificación se ha creado una base de datos de imágenes en cierto modo escasa si se compara con grandes proyectos de visión artificial. Pues bien, para este proyecto, el número de muestras máximo usado para la detección cada objeto ronda las 4000 imágenes, pero, si se desea mejorar las condiciones del detector el número de muestras puede ser ilimitado.

Dicho esto, es evidente que si se pretende mejorar el detector para este proyecto se debe ampliar de manera considerable el número de muestras a utilizar y también las vistas y cercanía para cada uno de los objetos. Otras formas de mejorar el detector es además del seguimiento por el método CamShift, añadir un filtro de Kalman que posibilite predecir la posición siguiente al "*frame*" inicial y que otorgue cierta estabilidad a las imágenes tomadas por la cámara. Por último, como es obvio, si se combinan o fusionan diferente sensores tales como cámaras térmicas o sensores de distancia los resultados serán mucho mejores.

El siguiente paso en este proyecto es comprobar a nivel de campo el sistema diseñado. En este paso el objetivo es conseguir embarcar los dispositivos electrónicos, esto es, la Pixhawk y la Raspberry Pi de manera que se aproveche de manera óptima el espacio disponible en la plataforma del dron.

Otro punto a destacar es colocar la cámara de tal modo que se obtenga el máximo campo de visión y que ningún objeto de interés pueda pasar desapercibido.

Una vez se haya realizado las pertinentes pruebas en campo es posible realimentarse para establecer rutas de operación con mayor grado de dificultad. Una posible solución que se baraja es conseguir que el dron siga la trayectoria del objeto intruso, esto es posible si se consigue monitorizar el centroide del objeto en cuestión y que el dron siga en todo momento el trazado de este punto.

# BIBLIOGRAFÍA


**[1] OpenCV: Detection Using Haar Cascade Disponible en: <https://opencv.org> [Abril 2018]**

**[2] Raspberry Pi Disponible en: <https://www.raspberrypi.org> [Abril 2018]**

**[3] Pixhawk Disponible: <https://pixhawk.org/modules/pixhawk> [Abril 2018]**

**[4]Ardupilot Disponible en: < http://ardupilot.org/copter/ > [Abril 2018]**

**[5] Vehicle Detection in Aerial Imagery: A small target detection benchmark., Sébastien Razakarivony and Frédéric Jurie, Journal of Visual Communication and Image Representation, 2015**

**[6] INRIA Person Dataset Disponible en: <http://pascal.inrialpes.fr/data/human/> [Marzo 2018]**

**[7] Jones and Viola. "Rapid Object Detection using a Boosted Cascade of Simple Features" 2001 Disponible en: <http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf > [Abril 2018]**

**[8] Conexión Raspberry Pi Disponible en: < https://vueloartificial.com/1-conexion-vuelo-controlado-mediante-raspberry-pi/> [Abril 2018]**

**[9] Algoritmo Viola and Jones Disponible en: <https://es.wikipedia.org/wiki/Detecci%C3%B3n_de_caras#cite_note-Viola_&_Jones-5> [Abril 2018]**

**[10] Histograms of Oriented Gradients for Human Detection Disponible en: < https://hal.inria.fr/file/index/docid/548512/filename/hog_cvpr2005.pdf> [Abril 2018]**

**[11] Navneet Dalal, Bill Triggs. Histograms of Oriented Gradients for Human Detection. Cordelia Schmid and Stefano Soatto and Carlo Tomasi. International Conference on Computer Vision & Pattern Recognition (CVPR '05), Jun 2005, San Diego, United States. IEEE Computer Society, 1, pp.886–893, 2005.**

**[12]Tutorial sobre máquinas de vectores de soporte (SVM) Disponible en: < http://www.ia.uned.es/~ejcarmona/publicaciones/%5B2013-Carmona%5D%20SVM.pdf > [Abril 2018]**

**[13]Cascade Classifier Training Disponible en: < https://docs.opencv.org/3.3.0/dc/d88/tutorial_traincascade.html> [Abril 2018]**

**[14] A comparison of Haar-like, LBP and HOG approaches to concrete and asphalt**

**runway detection in high resolution imagery Disponible en: < http://epacis.net/jcis/PDF_JCIS/JCIS11-art.0101.pdf> [Abril 2018]**

**[15] GUEVARA, MARTA LUCÍA, ECHEVERRY, JULIAN DAVID, ARDILA URUEÑA, WILLIAM, DETECCIÓN DE ROSTROS EN IMÁGENES DIGITALES USANDO CLASIFICADORES EN CASCADA. Disponible en:<http://www.redalyc.org/articulo.oa?id=84903801> ISSN 0122-1701 [Mayo 2018]**

**[15] Robotics Operation System (ROS) Disponible en: <http://www.ros.org/about-ros/ > [Julio 2018]**

**[16] Uso civil aeronaves controladas por control remoto Disponible en: <https://www.boe.es/boe/dias/2017/12/29/pdfs/BOE-A-2017-15721.pdf> [Abril]**

**[17] Dron Disponible en: <https://www.buildyourowndrone.co.uk/dji-spreading-wings-s900-dji-a2> [Abril 2018]**

**[18]Estimated Costumer Drone Shipments Disponible en: <https://www.businessinsider.com/drone-industry-analysis-market-trends-growth-forecasts-2017-7?IR=T> [Julio 2018]**

**[19] DroneKit Disponible en:< http://python.dronekit.io/> [Marzo 2018]**

**[20] QGroundControl Disponible en: https://docs.qgroundcontrol.com/en/**

**[21] Python Disponible en: <https://www.python.org> [Abril 2018]**

**[22] Wikipedia Disponible en: <https://es.wikipedia.org/wiki/C%2B%2B> [Abril]**

**[23]Real VNC Disponible en: <https://www.realvnc.com/es/> [Abril 2018]**

**[25] Diapositivas Curso detección Objetos Disponible en: <https://d3c33hcgiwev3.cloudfront.net/_d89c679efd80817bfc9f5feacfa65be1_L2.2.c-LBP-HistogramaPorBloques.pdf?Expires=1535414400&Signature=BgEAXoHwfVb7J5ypsQyJPHvhoNeOWaeuZhmdGnbny~gThEJnLAuo00jAsrdKODkhOwcr3J1DEYZ3J85OR9PAGRvuo4W2rbyWqG54Bgj~RLEfdX8HxyUMq~9IAohqO4OEpweGLBNJh6D38M5YSQlKX4HsNz9koqjeCjIjuZEio4c_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A>[Mayo 2018]**

**[24]Curso Detección de Objetos Disponible en:<https://www.coursera.org/learn/deteccion-objetos/home> [Julio2018]**

**[26]Documentación OpenCV Disponible en: <https://docs.opencv.org/3.4.0/db/df8/tutorial_py_meanshift.html> [Julio 2018]**

**[27] Simulación PX4 Disponible en: <http://dev.px4.io/en/simulation> [Mayo 2018]**

**[28] Documentación Ardupilot Disponible en: <http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html> [Mayo 2018]**

**[29] Apuntes Asignatura Sistemas de percepción UC3M**